Module Guide for Software Engineering

Team #11, OKKM Insights
Mathew Petronilho
Oleg Glotov
Kyle McMaster
Kartik Chaudhari

January 18, 2025

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

See SRS Documentation at [https://github.com/OKKM-insights/OKKM.insights/tree/main/docs/SRS—SS]

Contents

List of Tables

List of Figures

3 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by Parnas et al. (1984), as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The specific hardware on which the software is running.

AC2: The format of the initial input data.

...

[Anticipated changes relate to changes that would be made in requirements, design or implementation choices. They are not related to changes that are made at run-time, like the values of parameters. —SS]

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

...

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Account Creation Interface

M3: Account Database Connector

M4: Account Database

M5: Account Update Interface

M6: Login Interface

M7: Access Token

M8: Labeler

M9: Client

M10: User

M11: Account Creation Controller

M12: Account Update Controller

M13: Authentication Controller

M14: Satellite Image Request Interface

M15: Satellite Image Request Controller

M16: Satellite Image Request

M17: Project Creation Interface

M18: Project Creation Controller

M19: Project

M20: Service Request Failure Interface

M21: Image Upload Interface

M22: Report Interface

M23: Report Controller

M24: Report

M25: Project Selection Interface

M26: Project Selection Controller

M27: Labeling Interface

M28: Labeling Controller

M29: Image

M30: Label Server

M31: Label Database Connector

M32: Label Database

M33: ImageObject Database Connector

M34: ImageObject Database

M35: Labeller Database Connector

M36: Labeller Database

M37: Object Extraction Manager

M38: Image Service Manager

M39: Label Confidence Service

M40: Object Extraction Service

M41: Image Prior Analyzer

M42: Labeller Expertise Calculator

M43: Image Mask Service

M44: Image Selection Service

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 2.

There are several key high level design decisions that were made to ensure the requirements outlined in the SRS are met. First, the system has been decomposed into several microservices. These services all connect to the same databases, but have the flexibility to operate on their own cadence. Each service can be mirrored by a duplicate service to improve capacity. These decisions are critical to ensuring the speed, uptime, capacity, and latency requirements. Each of the key algorithms in the system have been encapsulated in their own module. These include the object extraction service, image serving, ML model creation, and report generation, among others. This gives the development team the flexibility to experiment with alternative algorithms, allowing continuous improvement towards performance

and accuracy requirements. The authentication service is designed to meet the authentication requirements by ensuring only approved users can access privileged information. Every functional requirement is covered by the functionality of a subsystem.

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *Software Engineering* means the module will be implemented by the Software Engineering software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

7.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

7.2.1 Account Creation Interface (M2)

Secrets: The design format of the account creation user interface.

Services: Displays a form to collect user information and submits that information to be processed.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.2 Account Database (M4)

Secrets: The data structure, storage, and access mechanisms for account related data.

Services: Can store, insert, update and retrieve user account data.

Implemented By: OrbitWatch

Type of Module: Abstract Data Type

7.2.3 Account Database Connector (M3)

Secrets: The database access key and database access algorithms.

Services: Can request storage, insertion, updates and retrieval of user account data from the database.

Implemented By: OrbitWatch

Type of Module: Abstract Data Type

7.2.4 Account Update Interface (M5)

Secrets: The design format of the update account information user interface.

Services: Displays a form with the users current information that can be modified and submits any updated information to be processed.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.5 Login Interface (M6)

Secrets: The design format of the login user interface.

Services: Displays a form to collect login credentials and submits that information to be processed.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.6 Access Token (M7)

Secrets: Token structure, encryption, expiration, and renewal mechanisms.

Services: Can determine if a user's token has expired and allows the token to be renewed.

Implemented By: OrbitWatch

Type of Module: Abstract Data Type

7.2.7 Labeler (M8)

Secrets: The format and structure of Labeler data.

Services: Takes in the necessary data to create a Labeler.

Implemented By: OrbitWatch

Type of Module: Record

7.2.8 Client (M9)

Secrets: The format and structure of Client data.

Services: Takes in the necessary data to create a Client.

Implemented By: OrbitWatch

Type of Module: Record

7.2.9 User (M10)

Secrets: The format and structure of User data.

Services: Takes in the necessary data to create a User.

Implemented By: OrbitWatch

Type of Module: Record

7.2.10 Satellite Image Request Interface (M14)

Secrets: The design format of the interface for requesting satellite images.

Services: Displays a form to collect specifics on the satellite images, calculates estimated cost based on current form entries and submits the form information to be processed.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.11 Satellite Image Request (M16)

Secrets: The format and structure of the data needed for a satellite image request.

Services: Takes in the necessary data to create a satellite image request.

Implemented By: OrbitWatch

Type of Module: Record

7.2.12 Project Creation Interface (M17)

Secrets: The design format of the project creation interface.

Services: Displays a form to collect project details, calculates an estimated cost using these details and submits that information to be processed.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.13 Project (M19)

Secrets: The format and structure of the data needed for a Project.

Services: Takes in the necessary data to create a Project.

Implemented By: OrbitWatch

Type of Module: Record

7.2.14 Service Request Failure Interface (M20)

Secrets: The design format of the request failure interface.

Services: Displays a warning to users that something went wrong.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.15 Image Upload Interface (M21)

Secrets: The design format of the image upload interface.

Services: Allows users to upload images from their device, and validates they are images of the correct format.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.16 Report Interface (M22)

Secrets: The design format of the summary report interface.

Services: Displays statistics and results of a specific project.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.17 Report (M24)

Secrets: The format and structure of Report data.

Services: Takes in the necessary data to create a Report.

Implemented By: OrbitWatch

Type of Module: Record

7.2.18 Project Selection Interface (M25)

Secrets: The design format of the project selection interface.

Services: Display all available projects that a user can label images from.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.19 Labeling Interface (M27)

Secrets: The design format of the labeling interface.

Services: Displays an image to be labeled, displays label choices which can be selected, and allows images to be skipped.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.2.20 Image (M29)

Secrets: The format and structure of Image data.

Services: Takes in the necessary data to create a Image.

Implemented By: OrbitWatch

Type of Module: Record

7.2.21 Label Server (30)

Secrets: The steps to accept and store a label.

Services: Converts a label JSON object and passes it to be stored in the database.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.2.22 Label Database Connector (31)

Secrets: The steps to connect to and manipulate a database.

Services: Connects to a database and executes pushing and fetching of data.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.2.23 Label Database (32)

Secrets: The storing and retrieving of data.

Services: Allows connection by a database connector to facilitate the storage and retreival of data.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.2.24 ImageObject Database Connector (33)

Secrets: The steps to connect to and manipulate a database.

Services: Connects to a database and executes pushing and fetching of data.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.2.25 ImageObject Database (34)

Secrets: The storing and retrieving of data.

Services: Allows connection by a database connector to facilitate the storage and retreival of data.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.2.26 Labeller Database Connector (35)

Secrets: The steps to connect to and manipulate a database.

Services: Connects to a database and executes pushing and fetching of data.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.2.27 Labeller Database (36)

Secrets: The storing and retrieving of data.

Services: Allows connection by a database connector to facilitate the storage and retreival of data.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.2.28 Object Extraction Manager (37)

Secrets: The algorithms used to aggregate labels into usable ImageObjects for Model training.

Services: Calls other modules to efficiently compute the most likely set of objects from a set of independent labels.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.2.29 Image Service Manager (38)

Secrets: The algorithms used to select images to be served to a user.

Services: Calls other modules to select the next images to be server to a user.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

7.3.1 Account Creation Controller (M11)

Secrets: Form validation and account creation algorithms.

Services: Validates the form information, creates an account with that information, and can upload the account to the database.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.3.2 Account Update Controller (M12)

Secrets: Form validation and account modification algorithms.

Services: Gets current user information, validates the update form information and can pass account updates to the database.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.3.3 Authentication Controller (M13)

Secrets: Credential validation, access token validation, and access token generation algorithms.

Services: Validates credentials given by user, generates access tokens, and validates access tokens.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.3.4 Satellite Image Request Controller (M15)

Secrets: Form validation and image request algorithms.

Services: Validates the form information, creates and sends a request to a third party for the necessary pictures.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.3.5 Project Creation Controller (M18)

Secrets: Form validation and project creation algorithms.

Services: Validates project information provided and creates a new project.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.3.6 Report Controller (M23)

Secrets: Logic for getting statistics of a project and exporting images onto the users device.

Services: Get statistics and labeled images of project, and exports images to external devices.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.3.7 Project Selection Controller (M26)

Secrets: Logic for getting active projects and redirecting labelers upon selection.

Services: Gets valid projects that are currently available, and redirects labelers to a selected project labeling interface.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.3.8 Labeling Controller (M28)

Secrets: Algorithms for label creation, removal and submission.

Services: Creates label for an image, removes label from an image, and submits labels to be processed.

Implemented By: OrbitWatch

Type of Module: Abstract Object

7.3.9 Label Confidence Service (39)

Secrets: The algorithms used to determine confidence in detected ImageObjects.

Services: Calculates the confidence of a detected ImageObject.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.3.10 Object Extraction Service (40)

Secrets: The algorithms used to extract ImageObjects.

Services: Calculates the most likely ImageObjects from a set of labels and priors.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.3.11 Image Prior Analyzer (41)

Secrets: The algorithms used to calculate ImagePriors.

Services: Calculates the prior liklihood of a pixel being relevant.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.3.12 Labeller Expertise Calculator (42)

Secrets: The algorithms used to calculate Labeller Expertise.

Services: Calculates the expertise of a labeller in a given class using previous labels.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.3.13 Image Mask Service (43)

Secrets: The algorithms used to modify a given image.

Services: Modifies an image in an attempt to improve labelling accuracy or efficiency.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

7.3.14 Image Selection Service (44)

Secrets: The algorithms used to select the next image to show a user.

Services: Selects an image to show to a given user.

Implemented By: [OrbitWatch]

Type of Module: [Abstract Object]

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Table 2: Trace Between Requirements and Modules

Req.	Modules
FR0	M11, M2, M3, M4, M10, M9
FR1	M13, M6, M7, M3, M4, M10, M9
FR2	M13, M7, M3, M4, M12, M5, M10, M9
FR4	
FR5	
FR6	
FR7	
FR8	
FR9	M11, M2, M3, M4, M10, M9
FR10	M13, M6, M7, M3, M4, M10, M8
FR11	M13, M7, M3, M4, M12, M5, M10, M9
FR13	M27, M28, M??, M30, M32, M31, M38, M44
FR14	M22, M23, M37, M39, M40, M41, M43
LF0	M5, M6, M2, M14, M20, M17, M21, M22, M25, M27

Continued on next page

Req.	Modules
LF1	M5, M6, M2, M14, M20, M17, M21, M22, M25, M27
LF2	M5, M6, M2, M14, M20, M17, M21, M22, M25, M27
UH0	M5, M6, M2, M14, M20, M17, M21, M22, M25, M27
UH1	M5, M6, M2, M14, M20, M17, M21, M22, M25, M27
UH2	M5, M12
UH6	M27
UH7	M20
UH9	M27
PR0	M2, M11, M9
PR1	M2, M11, M8
PR2	M22, M23, M24
PR3	M27, M28, M29, M38, M44
PR5	M22, M23, M37, M39, M40, M41, M43
PR6	Related to all modules
PR7	Related to all modules
PR8	M21, M14, M15, M16, M43, M38, M44, M41, M34, M33,
PR9	Related to all modules
OE0	Related to all modules
OE1	M14, M15, M16
OE4	M??, M??, M??, M??, M??
OE5	M34, M33, M??, M??, M??, M??
MR0	Related to all modules
SE0	M13, M7, M27, M28, M25, M26
SE1	M13, M7, M17, M18
SE2	M2, M11
SE3	M2, M11
SE4	M20
SE5	M11, M12, M18, M37, M39, M40, M41, M43
SE6	
SE7	M4, M3, M34, M33, M35, M36, M31, M32, M??
SE8	M12, M11, M4
SE10	M3, M33, M35, M31

AC	Modules
AC1	M1
AC2	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

[The uses relation is not a data flow diagram. In the code there will often be an import statement in module A when it directly uses module B. Module B provides the services that module A needs. The code for module A needs to be able to see these services (hence the import statement). Since the uses relation is transitive, there is a use relation without an import, but the arrows in the diagram typically correspond to the presence of import statement. —SS]

[If module A uses module B, the arrow is directed from A to B. —SS]

Figure 1: Use hierarchy among modules

10 User Interfaces

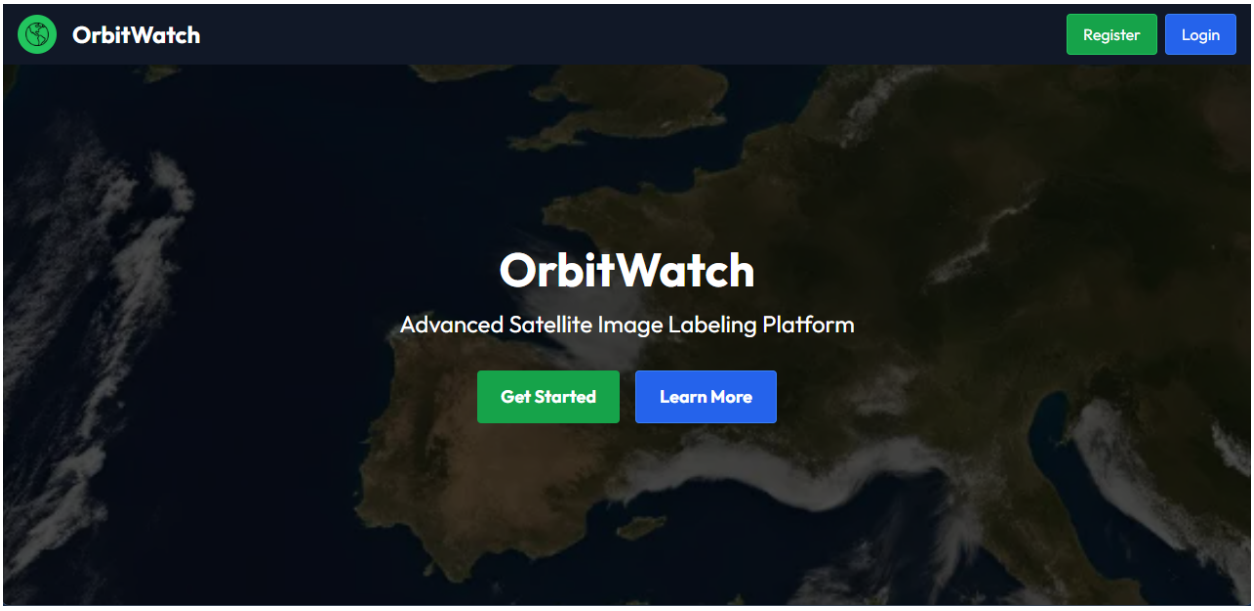

Figure 2: Home Page

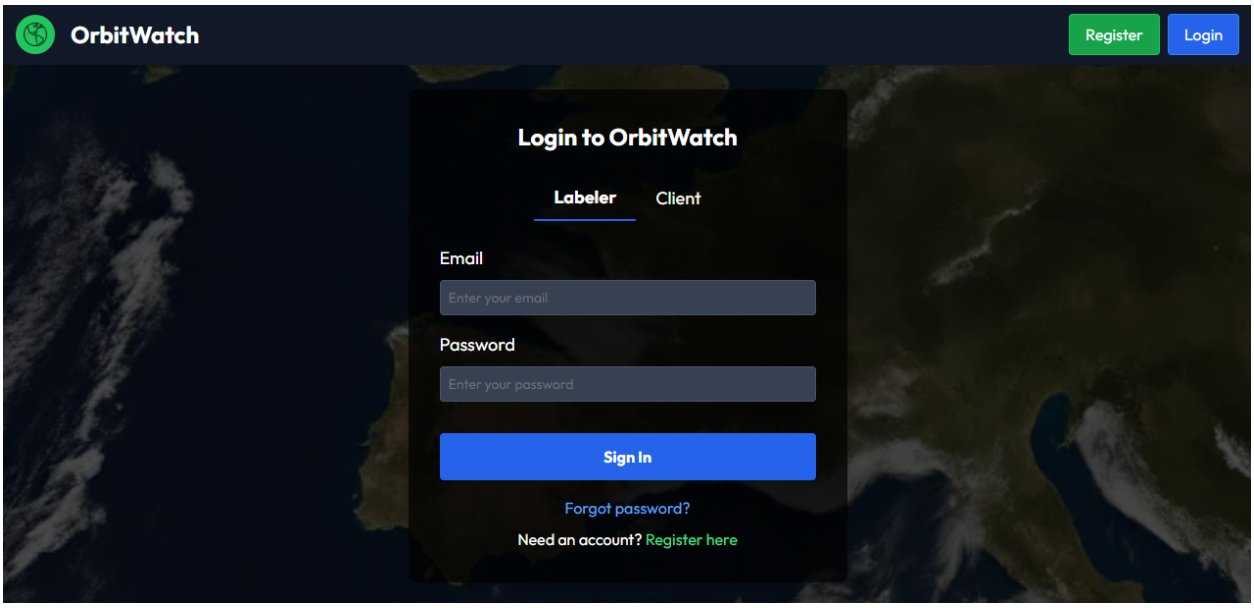

Figure 3: Login Page

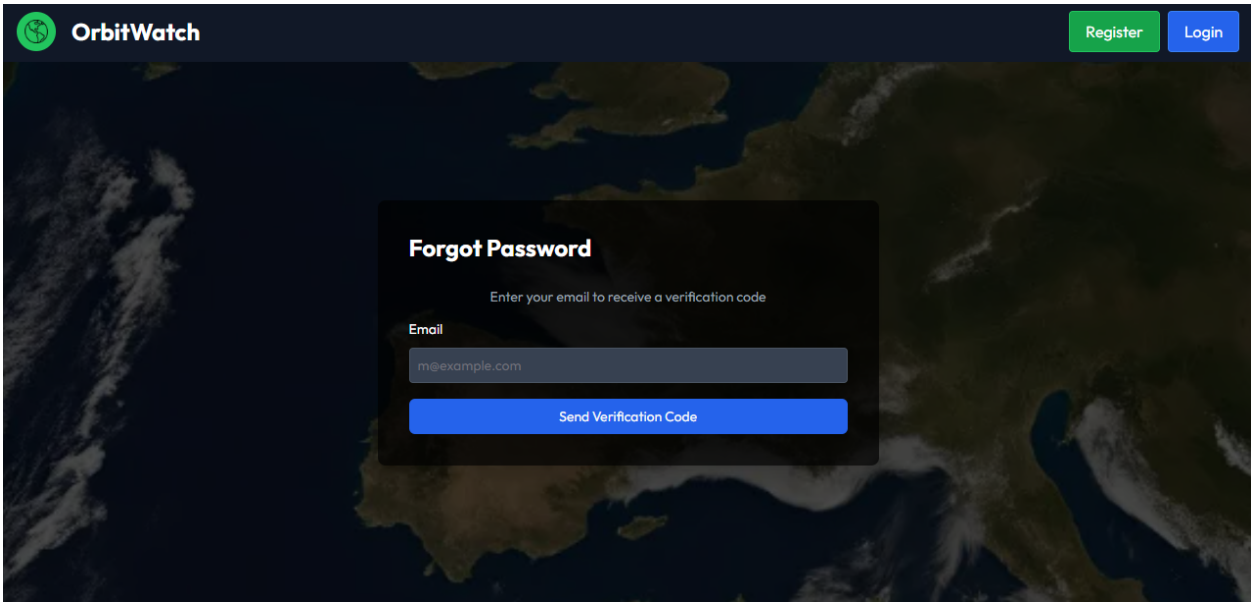

Figure 4: Forgot Password

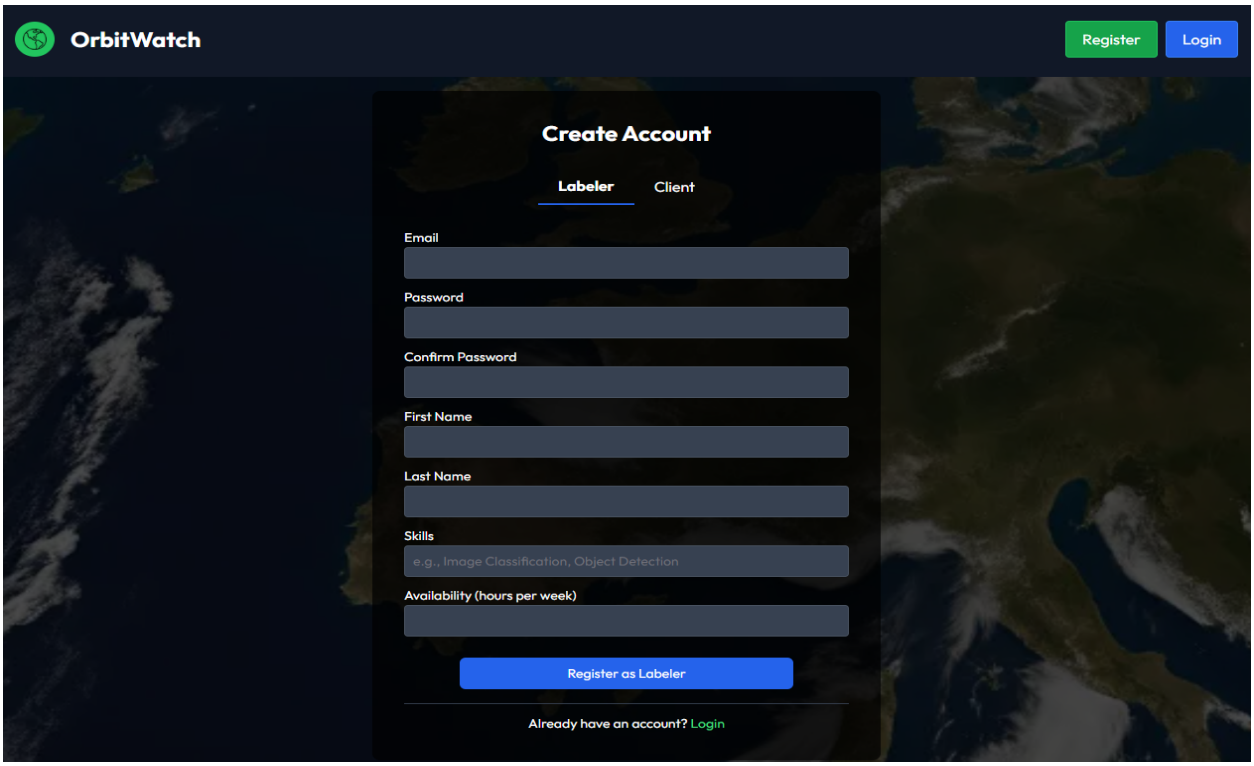

Figure 5: Register Page

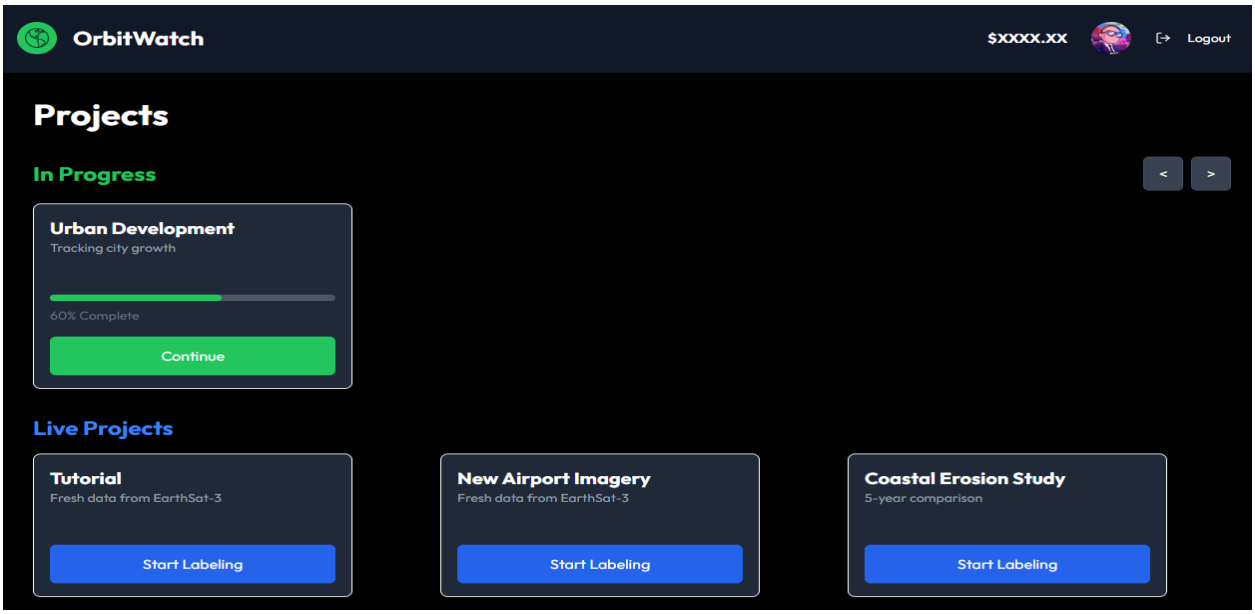

Figure 6: Projects to Label Page

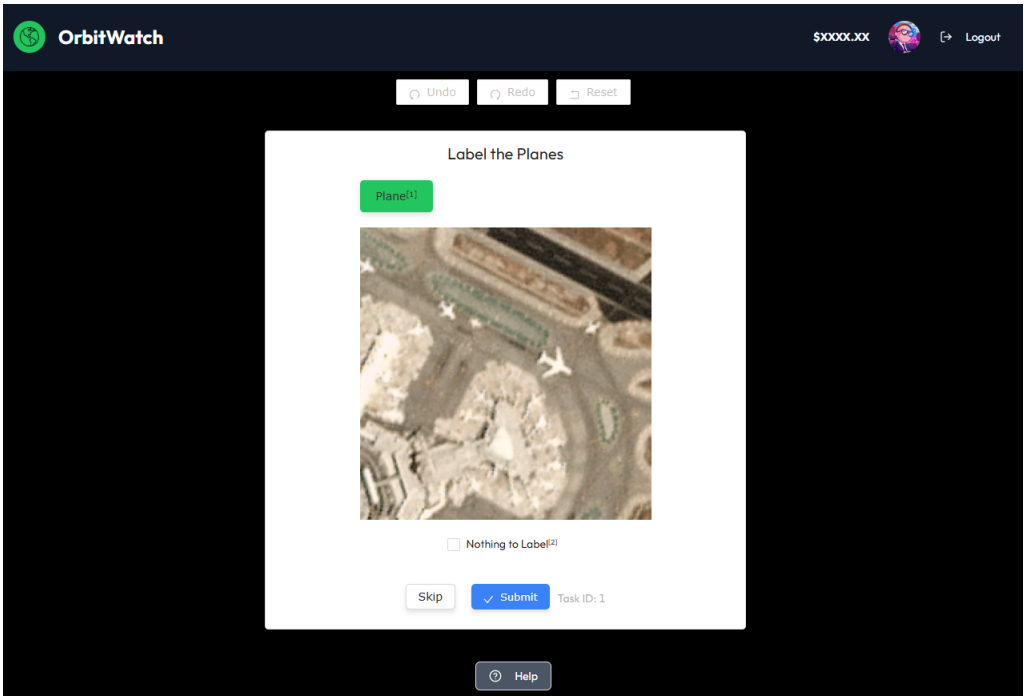

Figure 7: Label Page

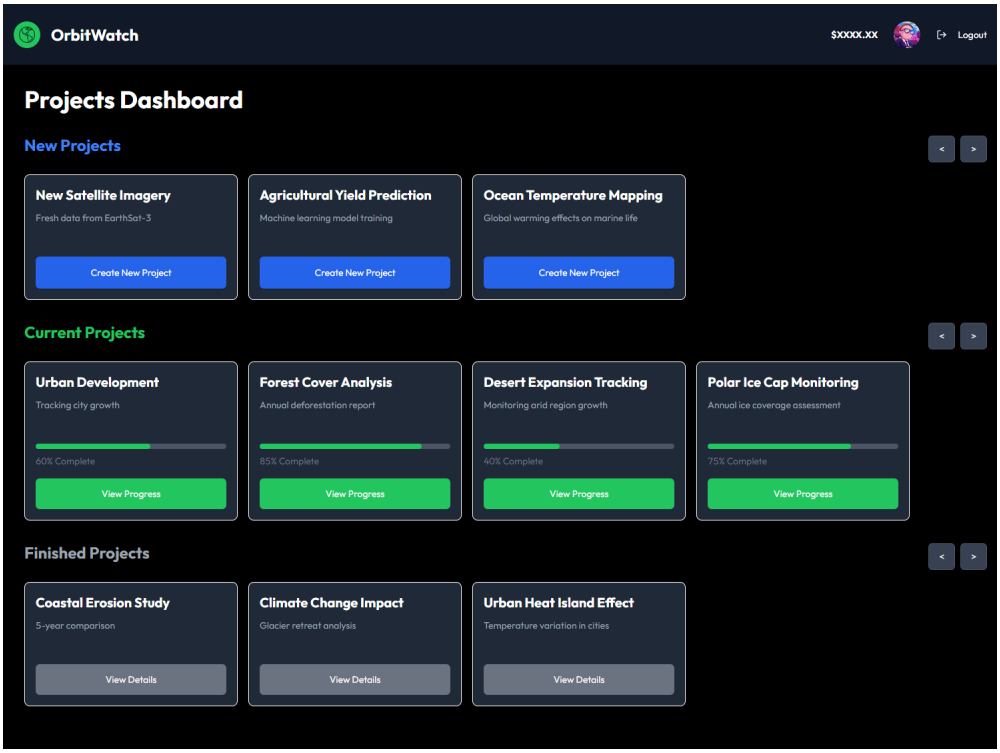

Figure 8: Client Projects Page

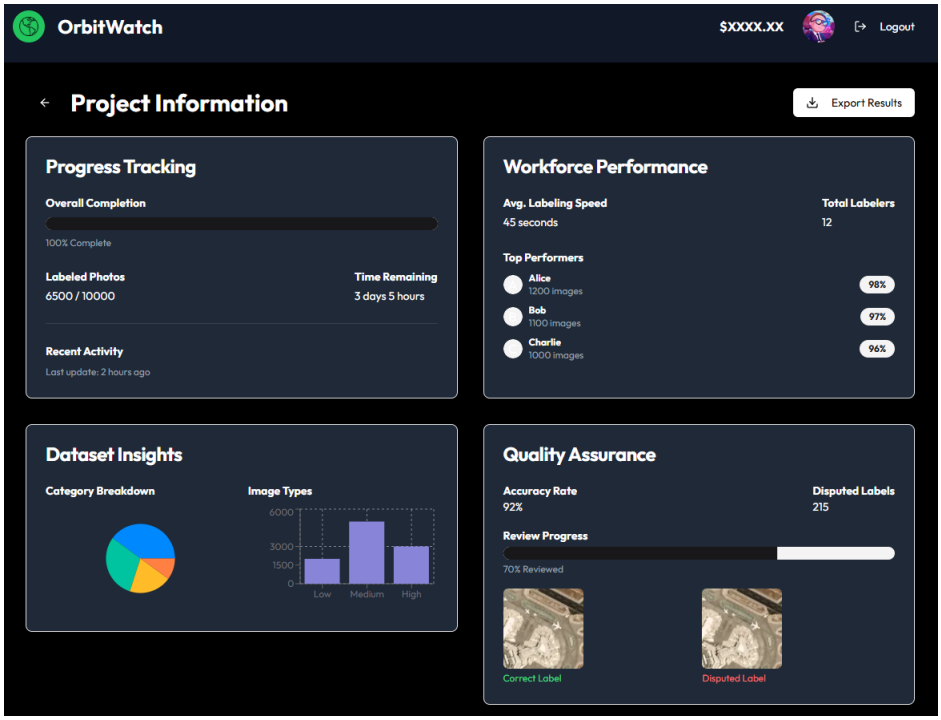

Figure 9: Project Status Page

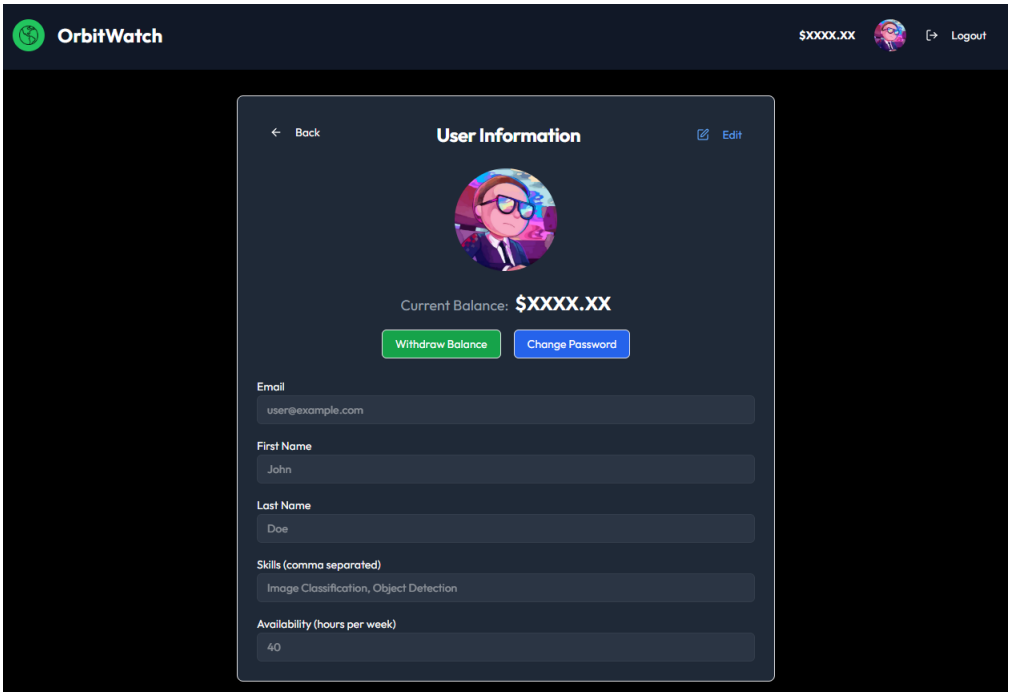

Figure 10: User Information Page

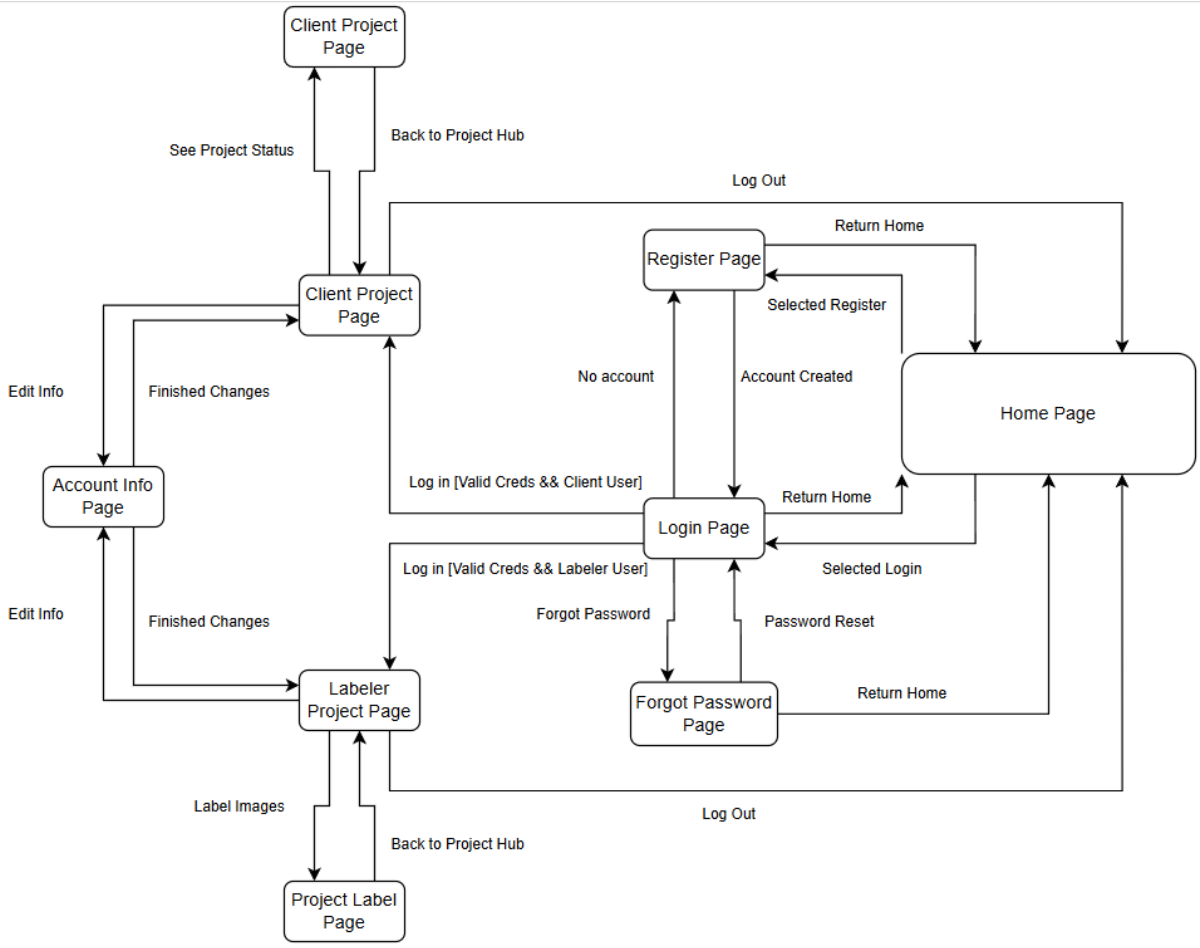

Figure 11: Page Flow Diagram

11 Design of Communication Protocols

[If appropriate —SS]

12 Timeline

[Schedule of tasks and who is responsible —SS]

[You can point to GitHub if this information is included there —SS]

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Account Creation Interface Account Database Account Database Connector Account Update Interface Login Interface Access Token Labeler Client User Satellite Image Request Interface Satellite Image Request Project Creation Interface Project Service Request Failure Interface Image Upload Interface Report Interface Report Project Selection Interface Labeling Interface Image Label Server Label Database Connector Label Database ImageObject Database Connector ImageObject Database Labeller Database Connector Labeller Database Object Extraction Manager Image Service Manager
Software Decision Module	Account Creation Controller Account Update Controller Authentication Controller Satellite Image Request Controller Project Creation Controller Report Controller Project Selection Controller Labeling Controller Label Confidence Service Object Extraction Service Image Prior Analyzer Labeller Expertise Calculator Image Mask Service Image Selection Service

Table 1: Module Hierarchy