Senior Design Project

"Bumpster"

High-Level Design Report

Supervisor:	İbrahim Körpeoğlu
Jury Members:	Bülent Özgüç Uğur Güdükbay
Innovation Expert:	Armağan Yavuz
Project Website:	www.bumpster.me
Group Members:	Duygu Durmuş Gülsüm Güdükbay Doğukan Yiğit Polat Muhammed Çavuşoğlu Bora Bardük

December 22, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

1.1 Purpose of the system

The system will generate physically based rendering maps using a specifically trained neural network. It will be able to identify the topological properties of the surface, as well as the material properties using a single photograph. It is supposed to provide rather large benefits for the industry, as the previous solution to this kind of topographical analysis required expensive methods which require personnel with expertise, expensive equipment, and time.

1.2 Design goals

Bumpster is developed based on several design goals which can be determined benefiting from nonfunctional requirements of the application. The application aims to reach these design goals given below.

1.2.1 Reliability

If there is a failure in the system, Bumpster should give an error to user related to the problem. Same error message should also be given to the developer in order to fix the problem as soon as possible. In addition, Bumpster should handle all of special cases which can cause crashes in the software. Especially, NULL values in the database should be considered for coping with them correctly. With the help of error handlers for the server, the software can minimize the number of failures.

Also, reliability of the information provided by Bumpster is important for application to help users with their generated maps. There are several existing systems based on texturing software which are good at exporting different kinds of maps from photographs or scanned images. However, the accuracy of results is not high. Bumpster should be an alternative to these existing systems for users searching for more reliable and accurate map results.

1.2.2 Availability

Similar existing systems are not available on mobile platforms so Bumpster should reach wide range of customers. Bumpster should be a multi-platform compatible application. It should be available on Google Play Store and App store. For android and iOS devices, users can download application Google Play Store and App store freely. For desktop use of application, the application should also be free and it should be available on http://bumpster.me/ for downloading. Hence, any user who has desktop, android device or iOS device can easily access and benefit from capabilities of the application.

1.2.3 Usability

Since the application targets wide range of audience between age groups of 18-45, the application should be easy to learn for audience. Although the target audience is capable of utilizing technology, the user interface should be easy and understandable for user-friendliness. The user interface should ensure easy tweaking of the preferences of generated map and view it from different angles. Hence, user interface and design for desktop and mobile applications should be kept clear and simple which avoids user loss due to non-friendly user interface. The application aims to provide and collect information in the simplest way for usability and user friendliness of the system.

1.2.4 Portability

The application should be able to used in different platform. Especially in the case of portability, different from similar existing systems, Bumpster should have an mobile version of the application on Android and iOS. Since users should access application with mobile devices, there is no portability issue related to the system.

1.2.5 Performance

The system should be efficient and give fast response to user request. Software should give answer to peer request as fast as possible to provide best possible response and move on to process future requests in time. This efficiency in performance will be possible with the efficient algorithms and connections used between client and the server. In addition, the database will be used efficiently which enables easy inserting, deleting and updating in terms of time cost.

1.2.6 Extensibility

Bumpster should be compatible with future versions of application. Bumpster should be open to development and it should be easy to add and enhance features and functionalities of the system. The application is based on generating maps from photographs or scanned images so users such as graphical designers expect to obtain realistic maps. Since their expectations from these kinds of applications increase, enhancing the existing system or new editing features might be required in the future versions with new tools to keep up with their needs and make their lives easier.

1.2.7 Maintainability

Since Bumpster aims to be extensible, the system should be keep up with changes and meet new requirements. In addition, the operating systems such as iOS, Android, Windows, macOS and Linux always improve via their updates. The application will be able to deal with the operating systems by the aid of its developers adding updates and also via the fact that the application is cross platform. Also, the data about users should be maintained properly which can be possible with a systematic database.

1.2.8 Scalability

The software should handle growing amount of work. In this case, it should be prepared for increasing number of users. In order to do this, the software show grow and enhance its functionalities to meet new requirements. There might be possible change at hardware scale which will provide processing power on a larger scale. The server-client architecture will be able to supply the increasing demand.

1.3 Definitions, Acronyms and Abbreviations

Amazon S3: Amazon Simple Storage Service

API: Application Programming Interface

UI: User Interface

Authentication: Confirming users' identity

Client: Part of the application that users interact with

Server: Part of the application that controls data management, computational and logical operations.

1.4 Overview

Bumpster is a tool that extracts the topographical and material properties of a surface and models it using a single image as input. Bumpster uses a neural network specifically trained for this purpose. While tools to achieve what Bumpster does similarly exist on the market today, there is no tool to accomplish this using artificial intelligence. These tools use image processing to extract data, and they require multiple input images to be taken in a studio environment with specific lighting conditions. Using machine learning enables Bumpster to differentiate itself from its competitors by giving users the freedom to input any image they wish to be topographically modeled. It can be used to capture real life situations and can be used to generate a topographical models instantly. This will benefit especially independent artists and small game developers since they may not have access to expensive tools, people with expertise, and a large amount of time. While this is the main concept, the final product will have a cross-platform front-end application with a user interface to send data to the server containing the neural network, and manipulate the model information retrieved from the server.

While Bumpster is a powerful data extraction tool, it is also a successful model manipulation environment on various platforms. After the extraction of the user selected mapping, the user will be able to obtain the resulting map and fit it onto the original photo to see the material properties or bump to create a realistic image. Then, the artists can tweak the chrominance, roughness and other properties of the final product and save it for later usage. Our team with professional background on platform specific app development will integrate hardware and software optimizations for the final product to work seamlessly.

2. Current software architecture

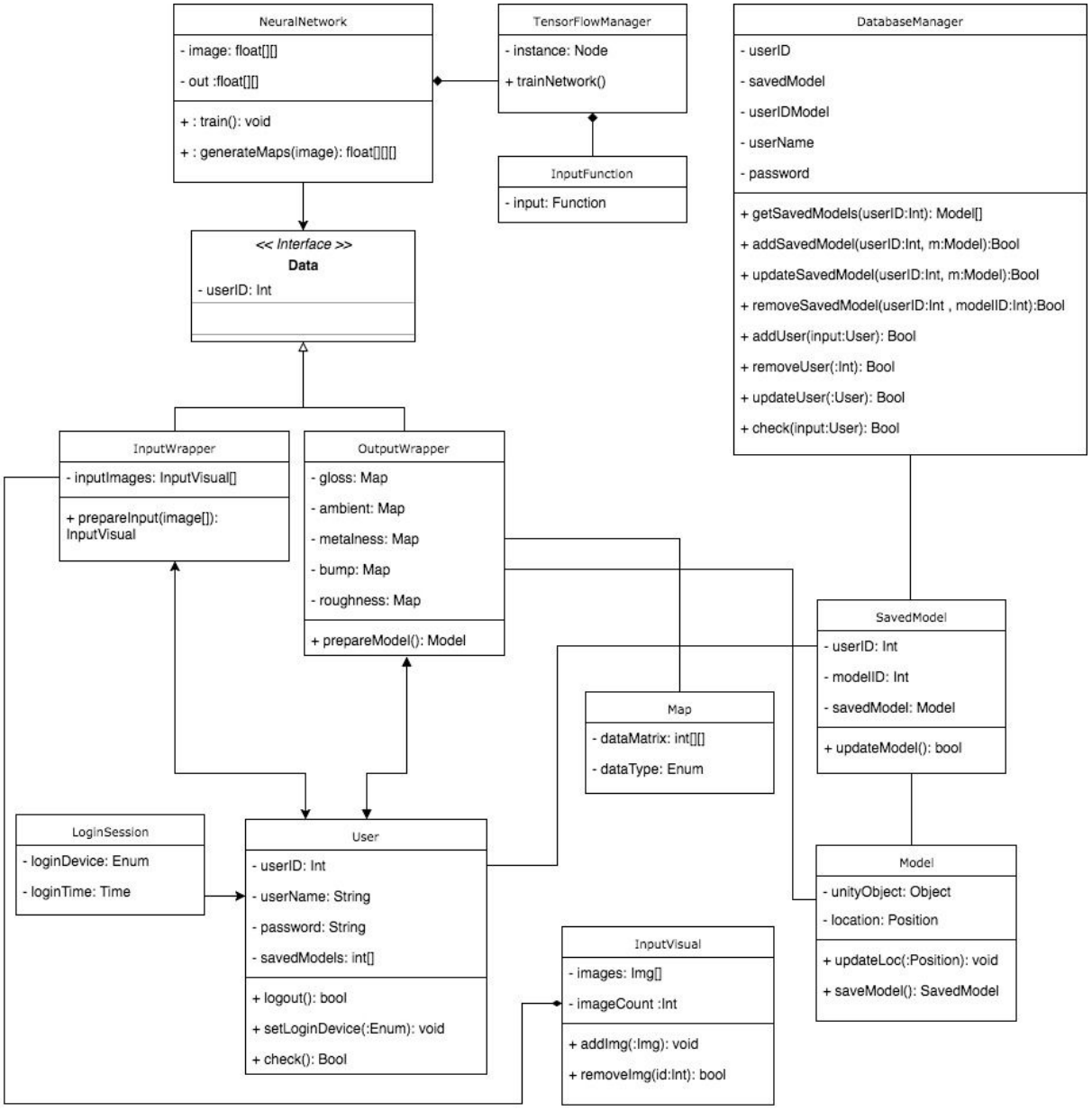

Figure 1 - UML Diagram for Object/Class Relationships

In our current architecture there will be two main parts cooperating in order to serve to user's requests. One of them is the back-end system that runs the machine learning models and generates the proper outputs for corresponding user requests. The other one is the client-side application which enables a user to send images to the back-end system and retrieve the output in a convenient manner.

3. Proposed software architecture

3.1 Overview

Bumpster aims to generate bump and physically based rendering maps using trained neural network from uploaded photographs or scanned images by the user. Required operations for identifying topological and material properties of the surface are sustained in the server. Then, back-end tier communicates with the front-end tier to respond requests of users on various platforms. It involves both client and server system with a connection to database.

In this case, the most suitable software architecture type for Bumpster to capture the runtime elements and interactions between them is client-server style as a component and connector view. It is possible to emphasize the interactions of components which are especially through synchronous invocation of capabilities. Since Bumpster also includes many components requesting services from other components and providing services to other components, transferring the component service can be explained in detail with client-server style of C&C view.

In addition, for the subsystem decomposition, layered style of module views is the best choice for describing the software architecture of Bumpster. It is because layered style reflects a division of software into layers. Each layer consists of group of modules working together to provide a set of services. Bumpster can also be divided into layers based on allowed-to-use relationship for separation of concerns as an advantage.

3.2 Subsystem Decomposition

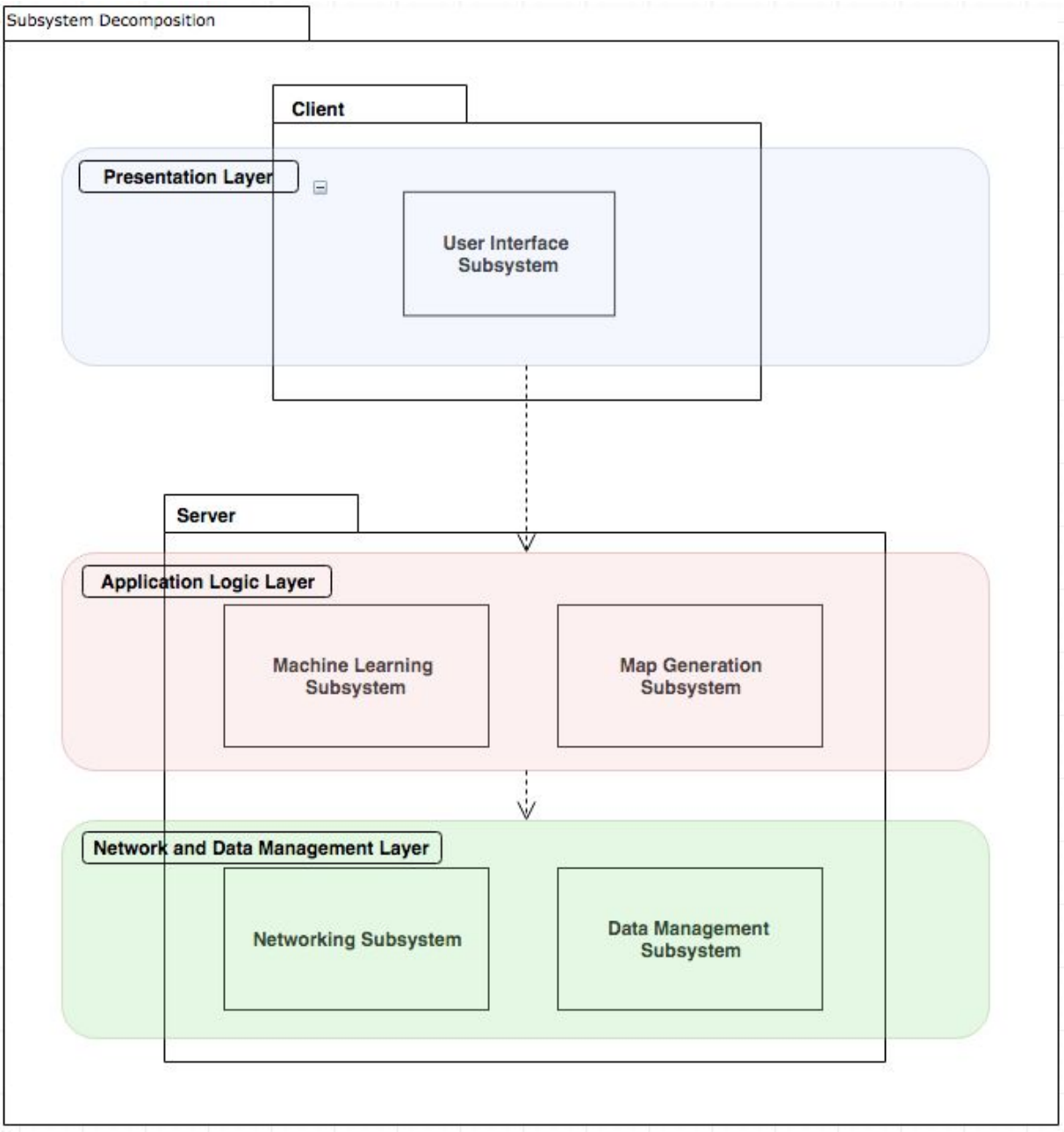

Figure 2 - Subsystem Decomposition

3.2.1 Presentation Layer

The presentation layer contains user interface subsystem. It is the layer which responds to the request of users. Hence, presentation layer is the layer which interacts with the users. This layer is also a client since it requests services from machine learning subsystem and map generation subsystem found in application logic layer.

3.2.2 Application Logic Layer

The application logic layer consists of machine learning subsystem and map generation subsystem. Map generation subsystem benefits from machine learning subsystem for generating several types of maps. Therefore placing these subsystems into the same layer is logical. Both of subsystems provide services to the presentation layer so application logic layer can also be seen as a server. Also, this layer uses network subsystem and data management subsystem found in the network and data management layer.

3.2.3 Network and Data Management Layer

Network and data management layer consists of networking subsystem related to issues such as user account management and client-server communication and data management subsystem related to manager and model classes for insertion, editing and deletion of users' models. This layer is also a part of server because it provides these services explained above to clients.

3.3 Hardware/Software Mapping

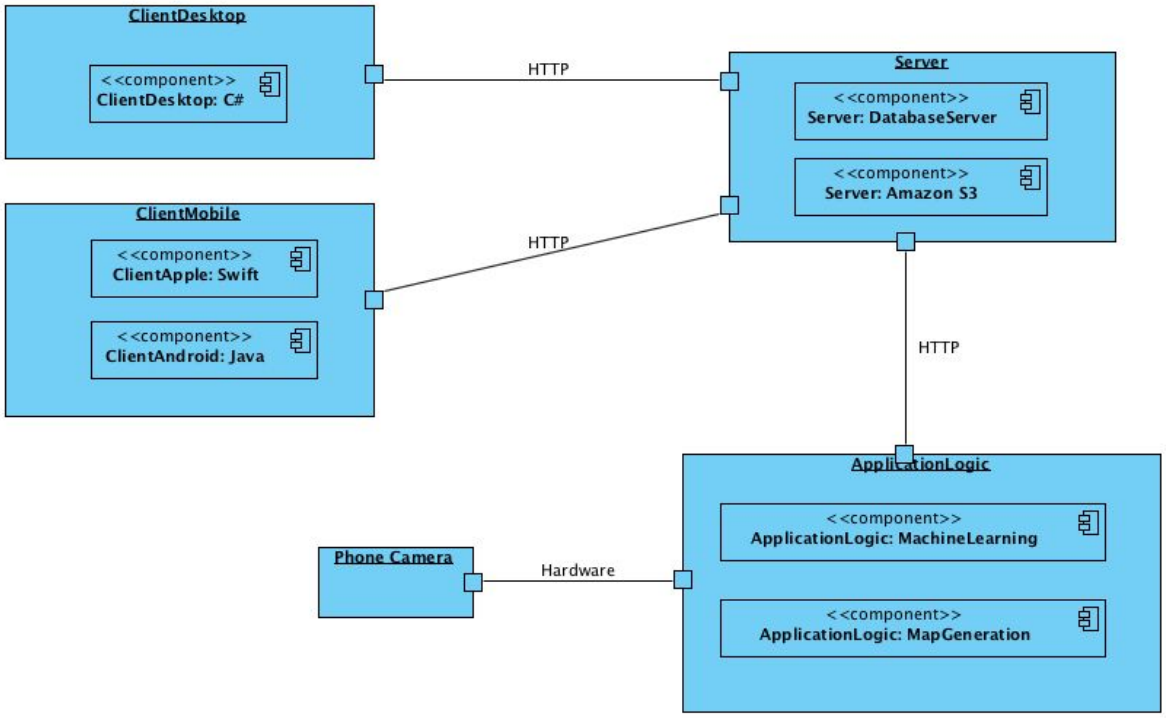

Figure 3 - Hardware Software Mapping of Bumpster

For the above hardware software mapping diagram, we used a component and connector view and this architecture is Client and Server style. There are two clients: ClientMobile and ClientDesktop referring to the two mobile applications that are for Apple and Android phones and a C# client for desktop. These include the GUI for usage of the application. The Server includes the database server which is MySQL and which has the user information and the Amazon S3 server that has the users' models saved. The clients and the server is connected by HTTP. The application logic includes the Machine Learning and Map Generation layers. It is connected to the server via HTTP. The only hardware in this application is the phone camera that allows the users to take their photo(s) and send them to Bumpster for the desired physically based rendering map generation. The phone will be connected to the server that will allow the phone to send the photo to the server part and to be used in the ApplicationLogic component.

3.4 Persistent Data Management

Users of the application will be able to send taken photos to the server and retrieve models generated by our application. In order to achieve that, we are going to use Amazon S3 for object storage. Amazon S3 provides flexible set of storage management and administration capabilities. We can classify, report and visualize usage of the data to optimize our application. It also offers features such as Backup & Recovery which we can use to restore users generated models to ensure that they are not lost [1].

Photos taken by the user are going to be sent to S3, and models generated by our algorithms are going to be stored on and be ready for retrieval from S3. Only authenticated users will have access to their photos and models. We are planning to use Firebase Authentication in our application to authenticate our users.

Firebase Authentication provides multi-platform sign-in functionalities. In addition to email and password accounts, it supports Google, Twitter, Facebook and GitHub logins. It is available in multiple platforms including iOS, Android, Web and Unity. It provides a customizable UI flows for sign in [2]. In conclusion; we are going to use Amazon S3 for photo and model storage, Firebase Authentication to authenticate our users by storing email and passwords in Firebase.

3.5 Access control and security

Users need to be authenticated to use our systems. Authenticated users can send new photos for generating models, view their existing models, delete their models and share model files. Only authenticated users will have access to only their profile information.

We are going to use Amazon S3 for object storage, and it has comprehensive security capabilities. It supports security standards and compliance certifications, helping developers build secure applications and satisfy compliance requirements. It offers capabilities for logging and monitoring API call activities for auditing, it also uses machine learning to automatically discover, classify. and protect sensitive data stored in S3. It is trusted and used by leading companies in industry including Netflix, AirBnB and Zillow [1].

We are going to use Firebase Authentication to authenticate our users. Firebase Authentication is built by the team that developed Google Sign-in and Chrome Password

Manager, and applies Google's internal expertise of security practices. It is also trusted and used by industry leading companies including The New York Times, The Economist and Shazam [2].

Our application will have access to device camera and photo galleries, only after user allows the application to access camera and photo gallery. This will be achieved by a pop-up displayed in the app which is mandatory for any app that needs access to camera or photo gallery. Users personal information, other than email address and password for authentication, will not be collected by our system.

3.6 Global Software Control

Bumpster implements event-driven software control on most levels. But because our system consists of many sub actors with different nature, like a neural network which processes requests and a user interface pushing data to the server/manipulating data on user request, the use of data-driven models are also required.

3.6.1 Create User (Event-Driven)

Creating a user account in Bumpster is an externally generated event. When user wants to create an account, the user provides the system with the required information we require for registration in the user interface on user's own system. When user submits the data, the system's database is notified with the user registration request and runs the necessary queries for the procedure.

3.6.2 Login User (Event-Driven)

When user wants to login to an account, the user provides the system with the credentials in the user interface on user's own system. This software control is event-driven since it is an externally driven event. When user submits the credentials, the system's database is notified with the user login request and runs check queries to identify the user.

3.6.3 Generate Maps (Data-Driven)

For the system to generate maps, the user first needs to supply the system with input image(s). After users do so, the system's neural network controller is notified of this request and

schedules the request of the current user. While there are requests made and not processed, the neural network will be busy generating output. This nature of flow makes this activity data-driven. The system then returns user the map generated by the neural network.

3.6.5 View Models on Mobile (Event-Driven)

While the user asks to retrieve models to their preferred device, a request is sent to our data server, asking to retrieve the required data. This flow of events generated is event driven since an external stimuli starts the flow.

3.7 Boundary conditions

This section includes the boundary conditions such as initialization, termination and failure.

3.7.1 Initialization

Considering the application-side of our system, users on all platforms will be required to enter their credentials and login to the system. When they do so, they will be greeted with a welcome screen. If they fail, they will be denied access to their accounts and will be sent back to the login screen with warning indications. Because our application retrieves data from the neural network and database of users (since users can view their pre-generated models on all their devices from same account), a connection to the internet is required for operation.

3.7.2 Termination

The user can log out from their Bumpster accounts by following related user interface elements. If Bumpster front-end application running on any platform detects inactivity, it may automatically terminate the session to avoid redundant operation, but it will not log out from user's device to provide ease of use in their next session. In case of logging out, Bumpster will clear the user data from the device, and require credentials to be entered again once the application is attempted to be used again.

3.7.2 Failure

There are two types of failures perceived during the current project phase. One is when the back-end operations undertaking in our system exceeds the possible load, and force a shutdown. In this case, an automated or manual system check will run for the system to be restore, and problems will be identified. The other case would by the loss of connection to the server during data transfer. While Bumpster can save the generated data to their database servers to be retrieved later, the case where user sends data to the Bumpster database is volatile. It may results with a natural loss of data, and possible reinitiation of the current workflow of users.

4. Subsystem services

4.1 User Interface Subsystem

User interface subsystem will provide user a set of convenient and easy-to-use tools for pre-processing the input data and post-processing the output data. It is placed in presentation layer to interact with users. It displays available data to the users. In order to make the system less complex, this subsystem is separated from other subsystems. User interface provides a connection between computer graphics, image processing tools and graphic designers. This subsystem will ensure that an easy and understandable user interface for user-friendliness. Hence, users can edit generated maps and view them from different angles easily.

4.2 Machine Learning Subsystem

This subsystem will not run regularly together with other subsystems and will be used to train new models or tweak the existing models to power the map generation subsystem. Models must be compatible with the interface that Map Generation Subsystem uses. This subsystem is separated from other subsystems because combining machine learning subsystem with map generation subsystem will cause complex system structure and avoid enhancing subsystems independently. Hence, for the sake of separation of concerns, it is better to separate machine learning subsystem.

4.3 Map Generation Subsystem

Map generation subsystem will be able to generate the PBR maps using models provided by the Machine Learning Subsystem through a common interface. This subsystem is responsible for generation of maps which are then displayed to users and edited by them. There are several type of maps generated for each upload of photograph(s) or scanned images. The subsystem itself is also complex, so it is logical to consider map generation subsystem as a separate subsystem.

4.4 Networking Subsystem

This subsystem will handle user account management and client-server communication related issues. The user will send their input photo(s) of a surface and the photo(s) will be sent to the server for map generation. The server will then output the generated map(s) to the user.

4.5 Data Management Subsystem

This subsystem will provide data access management connecting the GUI with the database. It will include the database manager classes and the model classes and will enable the insertion, editing and deletion of users' models.

6. References

[1] “Amazon Simple Storage Service”. https://aws.amazon.com/s3/?hp=tile&so-exp=below. [Accessed: Dec. 17, 2017].

[2] “Firebase Authentication”. https://firebase.google.com/products/auth/. [Accessed: Dec. 17, 2017].