

MINOR PROJECT REPORT

On

“PHOTO CRAFT” REAL TIME PHOTO STORE & MAKE SOCIAL

Submitted to Kurukshetra University in partial fulfillment of the requirement for the
award of the degree of

B.TECH

in

COMPUTER SCIENCE & ENGINEERING

**Submitted By
ANIKET RAJ**

Roll. No. 2218201



**DEPTT. OF COMPUTER SCIENCE & ENGINEERING
SWAMI DEVI DAYAL INSTITUTE OF ENGINEERING & TECHNOLOGY,
PANCHKULA, HARYANA**

NOVEMBER 2019

Certificate

I hereby certify that the work which is being presented in B.Tech major project report entitled, “**Real time photo store & make social**” in partial fulfillment of the requirements for the award of Degree in **Bachelor of Technology in Computer Science and Engineering** submitted to the **Department of Computer Science and Engineering** of **Swami Devi Dayal Institute of Engg. & Technology, Panchkula (Haryana)** is an authentic record of my own work carried out during a period from August 2019 to November 2019 under the super vision of **Er. kamal kumar**.

The matter presented in the report has not been submitted by us for the award of any other degree elsewhere.

Signature of the Candidate:

Aniket Raj (2218201)

Signature of the guide:

Er. Kamal Kumar

Signature of HOD:

Er. Kamal Kumar

Declaration

I hereby certify that the work which is being presented in the Major Project Report entitled, “**Real time Store & make social**” by me in partial fulfillment of the requirements for the award of Degree in **Computer Science and Engineering** submitted in the **Department of Computer Science and Engineering** at **Swami Devi Dayal Institute of Engg. & Technology, Panchkula (Haryana)** is an authentic record of my own work carried out under the supervision of **Er. Kamal Kumar**. The matter presented in the report has not been submitted in any other Institute for the award of any degree.

Aniket Raj 2218201

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Er. Kamal Kumar

(HOD / Lecturer)

Department of Computer Science and Engg.

Swami Devi Dayal Institute of Engg.

Technology

The Minor Project Viva-voce examination of **Amitabh Anand (2218201)** has been held on..... and is accepted.

Countersigned By: (Head of Department)

Acknowledgement

The writing of this project report has been assisted by the generous help of many people. I feel that I was very fortunate to receive assistance from them. I wish to express my sincere appreciation to them.

First and foremost, I am indebted to my supervisor, **Er. Kamal Kumar**(HOD/ Lecturer, department of Computer Science and Engineering) of Shivalik Institute of Engg. & Technology, who has been very supportive at every stage of my project completion. I wish to express my utmost gratitude to him for the invaluable advice and patience in reading, correcting and commenting on the drafts of this report and, more importantly, for his generosity which I have received throughout my project completion.

I would like to acknowledge and extended my heartfelt gratitude to **Er. Kamal Kumar**who helped and encouraged me throughout this journey.

I wish to express my thanks to all staff members of **Swami Devi Dayal Institute of Engg. & Technology**, who also helped me in conducting this study.

Finally, I am particularly indebted to my dearest parents/guardians as without their generous assistance and love; this project could never have been completed.

Aniket Raj

2218201

ABSTRACT

The objective of the project is to design a **Real time photo store & make social** application which enables the customers to create and manage a social platform of files. The project has been designed in **Node Js** technology and consists of **Disk Storage** database for the project.

Nodejs is one of the fastest growing technologies in today's world. The **Real time photo store & make social** project mainly consists for those types of users. The customers who access the information provided by the application and the creators who modifies and updates the information available in the application. All the data needed for the application is stored in the form of collection tables in the **Disk Storage** database.

CHAPTER 1

INTRODUCTION

1.1 Objective

The title of the project “**PHOTO CRAFT**” is a Social Web Application, where anybody can share, manage, design their data. Data can be in any form of media (Audio, video, images, docs, etc). The main objective of this project is enhancement of the social media in one place for all the socially active user and the developers too. It is the file storage and synchronization service. “**PHOTO CRAFT**” allows users to store files on their servers, synchronize files across devices, and share files.

Our Project covers the Foundation’s treatment of aggregate information collected by the Sites and personal information that you provide in connection with your use of the Sites. This Policy does not apply to the practices of third parties that the Foundation does not own or control, including but not limited to third party services you access through the Foundation, or to individuals that the Foundation does not employ or manage.

- CLOUD
- FILE MANAGEMENT (IN REAL-TIME)

Main objective to built or develop this project is to manage the file system in real time and connect or use it as cloud database. As an active users and developers can be able to upload and create their file at instance of a time when they needed. It will helpful for the end users to manage their folders as well as their files at one place.

It specifically designed to keep your data safe, secure and in your control. Your data belongs to you, and our tools enable you to control it, including who you share it with and how you share it anytime.

Search by keyword and filter by file type, owner and more. Google Drive can even recognize content in your scanned documents and images.

1.2 Problem Definition

To use facilities like share, save, manage, store and socially active are not in one place. Users have to use different web applications. If any user wants to be active on social media but along with this they want some documents to share so they cannot access on any one platform. One's have to visit or browse different web applications like –

“Pinterest” is a web that operates a software system designed to discover information on the world wide web. In talking with Pinners, we found that some people only wanted recommendations, while others liked to curate their feed—and most people wanted both options.

“Dribbble” is an online community for showcasing user-made artwork. Dribbble is a community of designers answering that question each day. Web designers, graphic designers, illustrators, icon artists, typographers, logo designers, and other creative types share small screenshots (shots) that show their work, process, and current projects.

“Pixlar” is photo storage system and make it social for every one to see and make it highest rank in the social world of this is photos

1.3 Brief Description

As we describe some of the major problems for social active user and a developer. **“PHOTO CRAFT”** is the solution for all because here a user as well as the developer can access and enjoy the feature of this project. It allows the users to see their files from any Internet-connected computer. It also simplified some of

the most common tasks, such as clicking only once on a file to see recent activity or share the file. It can create a special folder on the user's account that is default folder (Root and Shared), the content of which are then synchronized to **Disk Storage**.

So, in “**PHOTO CRAFT**” a user don't need to visit various web sites to get their need, they can access in this web application. Privacy Policy “**PHOTO CRAFT**” (the "Foundation") is committed to protecting the privacy of its users. This Privacy Policy (or the “Policy”) applies to its websites (whether currently or in the future supported, hosted or maintained, including without limitation nodejs.org, the “Sites”) and describes the information the Foundation collects about users of the Sites (“users”) and how that information may be used.

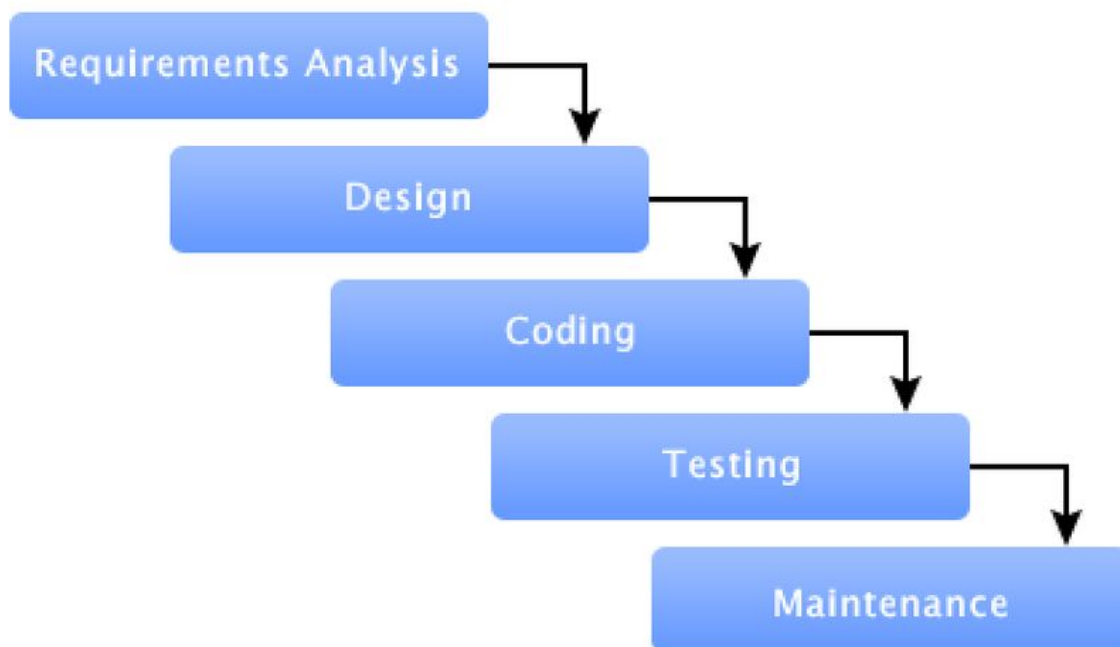
CHAPTER – 2

SOFTWARE DEVELOPMENT LIFECYCLE

2.1 Introduction

The software engineering process itself, for example, is usually divided into phases. The definition of these phases, their ordering, and the interactions between the phases specify a software life-cycle model. The best-known life-cycle model is the **Waterfall model** consisting of a

- Requirements Definition Phase
- Design Phase
- Coding Phase
- Testing Phase
- Maintenance



A software life cycle model is a descriptive and diagrammatic representation of the software life cycle. A life cycle model represents all the activities required to make a software product transit through its life cycle phases. It also captures the order in which these activities are to be undertaken. In other words, a life cycle model maps the different activities performed on a software product from its inception to retirement. Different life cycle models may map the basic development activities to phases in different ways. Thus, no matter which life cycle model is followed, the basic activities are included in all life cycle models though the activities may be carried out in different orders in different life cycle models. During any life cycle phase, more than one activity may also be carried out. For example, the design phase might consist of the structured analysis activity followed by the structured design activity.

Basically the term software lifecycle describes the development of an application, from the concept phase right up to the retirement phase. The purpose of such a plan is to define the various intermediate phases required to validate the development of the application i.e. to ensure the software conforms to the application and verification of development procedure, i.e. to make sure the methods employed are appropriate. Such plans originate from the fact that errors detected late in the implementation phase can end up being costly to rectify. The lifecycle allows for errors to be detected at as early a stage as possible and therefore enable developers to concentrate on the quality of the software, implementation time frames and associated costs.

The Basic software lifecycle involves the following procedure:

- Defining goals defining the outcome of the project, and its role in a global strategy.
- Analysis of requirements and feasibility, i.e. gathering, examine and formulating the customer's requirements and examining any restrictions that may apply.
- General design General architectural requirements of the application.
- Detailed design, precise definition of each application sub-set.
- Programming is the implementation of a programming language to create the functions defined during the design stages.
- Unit testing, individual testing of each application sub-set to ensure they are implemented according to specifications.

- Integration to ensure that the different modules integrate with the application. This is the purpose of the integration testing which is carefully documented.
- Beta testing or debugging, to ensure that the software conforms to original specifications,
- Documentation serves to document necessary information for software users and for future development.
- Implementation.
- Maintenance, all corrective procedures i.e. corrective maintenance and minor software updates i.e. ongoing maintenance.

The order and presence of each of these procedures in the lifecycle of an application depends on the type of lifecycle model agreed between the client and the development team. To facilitate a common methodology to both the client and the software company, lifecycle models have been updated to reflect the development stages involved and the documentation required, so that each stage is validated before moving on to the next stage. There are many software models which are followed by the industry according to their needs and requirement.

The waterfall model is a relatively linear sequential design approach for certain areas of engineering design. In software development, it tends to be among the less iterative and flexible approaches, as progress flows in largely one direction ("downwards" like a waterfall) to the phase initiation, analysis, design, construction, testing, deployment and maintenance.

The waterfall development model organized in the manufacturing and construction industries in where the highly structured physical environments meant that design changes became prohibitively expensive much sooner in the development process. When first adopted for software development, there were no recognized alternatives for knowledge-based creative work. In common practice, waterfall methodologies result in a project schedule with 20–40% of the time invested for the first two phases, 30–40% of the time to coding, and the rest dedicated to testing and implementation.

The actual project organisation needs to be highly structured. Most medium and large projects will include a detailed set of procedures and controls, which regulate every process on the project.

2.2 Feasibility study

2.2.1 Economic Feasibility

Our economic feasibility to maintain the active users interest. As user will always be able to make the decision whether to proceed with any activity that requests personal information including personally identifiable information. If a user does not provide requested information, the user may not be able to complete certain transactions.

- To understand a user's needs and create content that is relevant to the user.
- To generate statistical studies.
- To conduct market research and planning by send user survey;
- To notify user referrals of services, Information, or products when a user requests that the foundation send such information to referrals;
- To improve services, information, and products'
- To help a user complete a transaction, or provide services or customer support
- To communicate back to the user;
- To update the user on services, information and products;
- To personalize a site for the users.
- To notify the user of any changes with the site that may affect the user
- To enforce terms of use on a site and
- To allow the user to purchase product, access services or otherwise engage in activities the user selects

2.1.2 Technical Feasibility

User passwords are keys to accounts. Use unique numbers, letters, and special characters for passwords and do not disclose passwords to other people in order to prevent loss of account control. Users are responsible for all actions taken in their accounts. Notify the Foundation of any password compromises, and change passwords periodically to maintain account protection.

CHAPTER – 3

SOFTWARE REQUIREMENT ANALYSIS AND SPECIFICATION

3.1 SRS of the Project

Software requirement specification for this project consider the unique languages which are now going to trending in computer science field. Before the development of this project many languages are there which could have choose at that time but then after keeping in our mind we prefer to use some of these technologies i.e.

3.1.1 Node.js

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side. Historically, JavaScript was used primarily for client-side scripting. In which written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript for server-side scripting running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for a server side and client side scripts.

Though .js is the conventional filename extension for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web application (e.g. real time communication programs and browser games)

As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently.

Upon each connection the call back is fired, but if there is no work to be done, Node will sleep. This is in contrast to today's more common concurrency model where OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node are free from worries of dead-locking the process, since there are no locks. Almost no function in Node directly performs I/O, so the process never blocks. Because nothing blocks, scalable systems are very reasonable to develop in Node.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

3.1.2 Releases

The core team defines the roadmap's scope, as informed by Node.js' community. Releases happen as often as necessary and practical, but never before work is complete. Bugs are unavoidable, but pressure to ship a release will never prevail over ensuring the software is correct. The commitment to quality software is a core tenet of the Node.js project.

Patch releases:

- Include bug, performance, and security fixes.
- Do not add nor change public interfaces.
- Do not alter the expected behavior of a given interface.
- Can correct behavior if it is out-of-sync with the documentation.
- Do not introduce changes which make seamless upgrades impossible .

Minor releases:

- Include additions and/or refinements of APIs and subsystems.
- Do not generally change APIs nor introduce backwards-incompatible breaking changes, except where unavoidable.
- Are mostly additive releases.

Major releases:

- Usually introduce backwards-incompatible, breaking changes.
- Identify the API Node.js intends to support for the foreseeable future.
- Require conversation, care, collaboration and appropriate scoping by the team and its users.

3.1.3 Scoping Features

The team can add features and APIs into Node.js when:

- The need is clear.
- The API or feature has known consumers.
- The API is clean, useful, and easy-to use.

If when implementing core functionality for Node.js, the team or community may identify another lower-level API which could have utility beyond Node.js. When identified, Node.js can expose it for consumers.

For example, consider the `EventEmitter` interface. The need to have an event subscription model for core modules to consume was clear, and that abstraction had utility beyond the Node.js core. It was not the case that its interface couldn't be implemented externally to Node.js; instead, Node.js needed the abstraction for itself, and also exposed it for use by Node.js consumers.

Alternatively, it may be that many in the community adopt a pattern to handle common needs which Node.js does not satisfy. It may be clear that Node.js should deliver, by default, an API or feature for all Node.js consumers. Another possibility is a commonly-used compiled asset which is difficult to deliver across environments. Given this, Node.js may incorporate those changes directly.

The core team does not take the decision lightly to add a new API to Node.js. Node.js has a strong commitment to backwards compatibility. As such, community input and conversation must occur before the team takes action. Even if an API is otherwise suitable for addition, the team must identify potential consumers.

3.1.4 Deprecation

On occasion, the team must deprecate a feature or API of Node.js. Before coming to any final conclusion, the team must identify the consumers of the API and how they use it. Some questions to ask are:

- If this API is widely used by the community, what is the need for flagging it as deprecated?
- Do we have a replacement API, or is there a transitional path?
- How long does the API remain deprecated before removal?
- Does an external module exist which its consumers can easily substitute?

The team takes the same careful consideration when deprecating a Node.js API as they do when adding another.

3.1.5 EXPRESS

Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js. This is a built-in middleware function in Express. It parses incoming requests with JSON payloads and is based on body-parser.

Returns middleware that only parses JSON and only looks at requests where the Content-Type header matches the type option. This parser accepts any Unicode encoding of the body and supports automatic inflation of gzip and deflate encodings.

A new body object containing the parsed data is populated on the request object after the middleware (i.e. req.body), or an empty object ({}), if there was no body to parse, the Content-Type was not matched, or an error occurred.

You define routing using methods of the Express app object that correspond to HTTP methods; for example, app.get() to handle GET requests and app.post() to handle POST requests. For a full list, see app.METHOD. You can also use app.all() to handle all HTTP methods and app.use() to specify middleware as the callback function (See Using middleware for details).

These routing methods specify a callback function (sometimes called “handler functions”) called when the application receives a request to the specified route (endpoint) and HTTP method. In other words, the application “listens” for requests that match the specified route(s) and method(s), and when it detects a match, it calls the specified callback function.

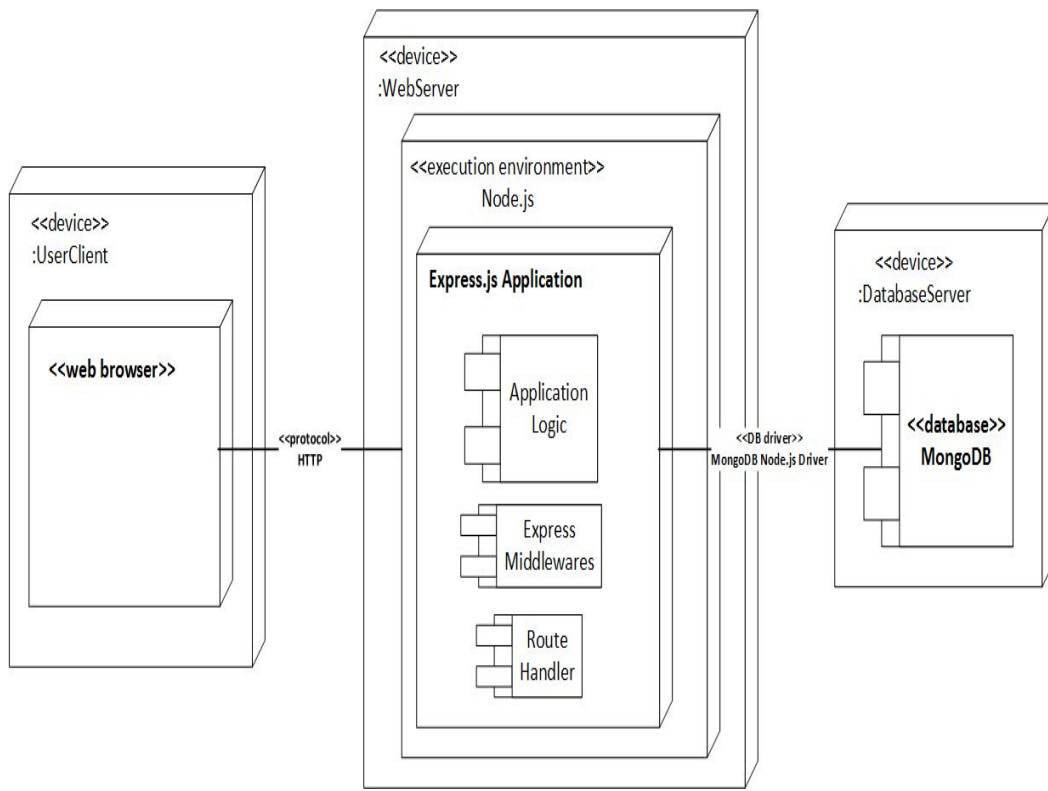
In fact, the routing methods can have more than one callback function as arguments. With multiple callback functions, it is important to provide next as an argument to the callback function and then call next() within the body of the function to hand off control to the next callback.

The following code is an example of a very basic route.

```
var express = require('express')
var app = express()

// respond with "hello world" when a GET request is made to the homepage
app.get('/', function (req, res) {
  res.send('hello world')
})
```

Working of EXPRESS



Middleware functions are functions that have access to the request object (req), the response object (res), and the next function in the application’s request-response cycle. The next function is a function in the Express router which, when invoked, executes the middleware succeeding the current middleware.

Middleware functions can perform the following tasks:

- Execute any code.
- Make changes to the request and the response objects.
- End the request-response cycle.

- Call the next middleware in the stack.

If the current middleware function does not end the request-response cycle, it must call `next()` to pass control to the next middleware function. Otherwise, the request will be left hanging.

3.2 Production Definition

“**PHOTO CRAFT**” abbreviation is “**STORE SHARE AND SHARE IT SOCIAL**”. Our project includes all the features of different web applications. In this web application we can create the heterogeneous file format which is having synchronous file management system in real-time as in any operating system. It provides us to create a platform where we can manage our data and socially connect with active user and share our resources among them. Here, we can manage our sharing file as a private or public.

STORE means a person whose job involves creative work and relating to or involving the use of the imagination or original ideas to create something. **producing or using original and unusual ideas.** having or showing an ability to make new things or think of new ideas. : using the ability to make or think of new things : involving the process by which new ideas, stories, etc., are created.

INNOVATIVE means the action or process of innovating. a person whose job involves producing original ideas or doing artistic work. The process of translating an idea or invention into a good or service that creates value or for which customers will pay. ... In business, **innovation** often results when ideas are applied by the company in order to further satisfy the needs and expectations of the customers.

SOCIAL means have a portion of (something) with another or others. o **share** something is to use it or enjoy it with others, like when people **share** a cake at a birthday party. The adjective **sharing** has a related meaning, but the focus is more on unselfishness. A **sharing** person might try to divide up not just the cake, but also all the food in his refrigerator — just to be sure everyone is fed.

Here, we connect all these things in one place i.e. “PHOTO CRAFT”.

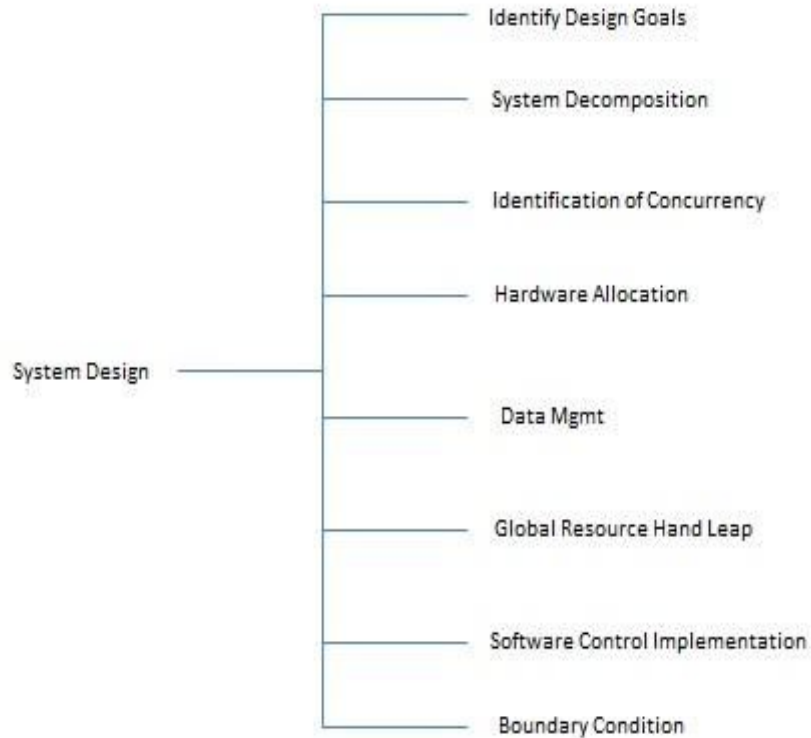
CHAPTER – 4

IMPLEMENTATION OF SYSTEM DESIGN

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain, i.e. *“how to implement?”*

It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate.

In this phase, the complex activity of system development is divided into several smaller sub-activities, which coordinate with each other to achieve the main objective of system development.



Inputs to System Design

System design takes the following inputs –

- Statement of work
- Requirement determination plan
- Current situation analysis

Types of Documentations

When it comes to System Design, there are following four main documentations –

- Program documentation

- System documentation
- Operations documentation
- User documentation

Program Documentation

- It describes inputs, outputs, and processing logic for all the program modules.
- The program documentation process starts in the system analysis phase and continues during implementation.
- This documentation guides programmers, who construct modules that are well supported by internal and external comments and descriptions that can be understood and maintained easily.

Operations Documentation

Operations documentation contains all the information needed for processing and distributing online and printed output. Operations documentation should be clear, concise, and available online if possible.

It includes the following information –

- Program, systems analyst, programmer, and system identification.
- Scheduling information for printed output, such as report, execution frequency, and deadlines.
- Input files, their source, output files, and their destinations.
- E-mail and report distribution lists.
- Special forms required, including online forms.
- Error and informational messages to operators and restart procedures.
- Special instructions, such as security requirements.

User Documentation

It includes instructions and information to the users who will interact with the system. For example, user manuals, help guides, and tutorials. User documentation is valuable in training users and for reference purpose. It must be clear, understandable, and readily accessible to users at all levels.

The users, system owners, analysts, and programmers, all put combined efforts to develop a user's guide.

A user documentation should include –

- A system overview that clearly describes all major system features, capabilities, and limitations.
- Description of source document content, preparation, processing, and, samples.
- Overview of menu and data entry screen options, contents, and processing instructions.
- Examples of reports that are produced regularly or available at the user's request, including samples.
- Security and audit trail information.
- Explanation of responsibility for specific input, output, or processing requirements.
- Procedures for requesting changes and reporting problems.
- Examples of exceptions and error situations.
- Frequently asked questions (FAQs).
- Explanation of how to get help and procedures for updating the user manual.

System Documentation

System documentation serves as the technical specifications for the IS and how the objectives of the IS are accomplished. Users, managers and IS owners need never reference system documentation. System documentation provides the basis for understanding the technical aspects of the IS when modifications are made.

- It describes each program within the IS and the entire IS itself.
- It describes the system's functions, the way they are implemented, each program's purpose within the entire IS with respect to the order of execution, information passed to and from programs, and overall system flow.
- It includes data dictionary entries, data flow diagrams, object models, screen layouts, source documents, and the systems request that initiated the project.
- Most of the system documentation is prepared during the system analysis and system design phases.

- During systems implementation, an analyst must review system documentation to verify that it is complete, accurate, and up-to-date, and including any changes made during the implementation process.

4.1 Model–view–controller

Model–view–controller (MVC) is an [architectural pattern](#) commonly used for developing [user interfaces](#) that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.^{[1][2]} The MVC design pattern decouples these major components allowing for efficient [code reuse](#) and parallel development

Descriptions

As with other software patterns, MVC expresses the "core of the solution" to a problem while allowing it to be adapted for each system .Particular MVC architectures can vary significantly from the traditional description here.

Components

- The *model* is the central component of the pattern. It expresses the application's behaviour in terms of the [problem domain](#), independent of the user interface. It directly manages the data, logic and rules of the application.
- A *view* can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- The third part or section, the *controller*, accepts input and converts it to commands for the model or view.

Installation MVC with express

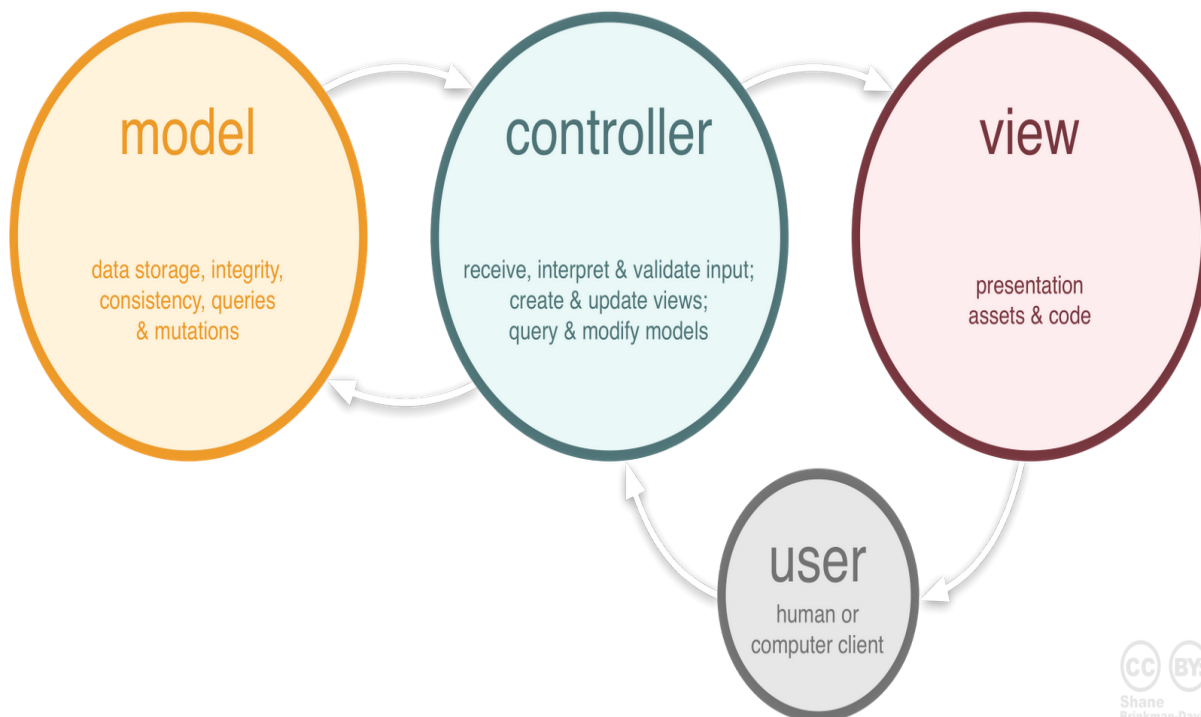
```
$ npm install express-mvc-generator -g.
```

Interactions

In addition to dividing the application into three kinds of components, the model–view–controller design defines the interactions between them.

- The model is responsible for managing the data of the application. It receives user input from the controller.
- The view means presentation of the model in a particular format.
- The controller responds to the user input and performs interactions on the data model objects. The controller receives the input, optionally validates it and then passes the input to the model.

4.2 Flow Diagram



Basic Diagram of Modal View Controller

4.3 REST API with express

REpresentational State Transfer (REST) is an architectural style that defines a set of constraints and properties based on [HTTP](#). Web Services that conform to the REST architectural style, or **RESTful web services**, provide interoperability between computer systems on the [Internet](#). REST-compliant web services allow the requesting systems to access and manipulate textual representations of [web resources](#) by using a uniform and predefined set of [stateless](#) operations. Other kinds of web services, such as [SOAP](#) web services, expose their own arbitrary sets of operations.

REST APIs are resource-based interfaces. On the web this means that data resources (typically formatted in JSON, or XML) are represented by URIs (paths) accessed via [HTTP](#).

Actions such as Create, Read, Update, and Delete (commonly referred to as [CRUD](#)) are made against these resources using HTTP methods (otherwise referred to as verbs): POST, GET, PUT/PATCH, and DELETE.

Each request made to a REST API is stateless. This means that the server handling your request maintains no context between requests: each request is interpreted equally. Any context required to process the request must be provided with the request itself (for instance, an authorization token).

"Web resources" were first defined on the [World Wide Web](#) as documents or files identified by their [URLs](#). However, today they have a much more generic and abstract definition that encompasses every thing or entity that can be identified, named, addressed, or handled, in any way whatsoever, on the web. In a RESTful web service, requests made to a resource's [URI](#) will elicit a response that may be in [XML](#), [HTML](#), [JSON](#), or some other format. The response may confirm that some alteration has been made to the stored resource, and the response may provide [hypertext](#) links to other related resources or collections of resources. When [HTTP](#) is

used, as is most common, the operations available are GET, POST, PUT, DELETE, and other predefined [CRUD HTTP methods](#).

4.3.1. CRUD OPERATION WITH MVC

In [computer programming](#), **create, read, update, and delete** (as an [acronym](#) **CRUD**) are the four basic functions of [persistent storage](#). Alternate words are sometimes used when defining the four basic functions of *CRUD*, such as *retrieve* instead of *read*, *modify* instead of *update*, or *destroy* instead of *delete*. *CRUD* is also sometimes used to describe [user interface](#) conventions that facilitate viewing, searching, and changing [information](#); often using computer-based [forms](#) and [reports](#). The term was likely first popularized by [James Martin](#) in his 1983 book *Managing the Data-base Environment*.^{[1][3]} The acronym may be extended to CRUDL to cover *listing* of large data sets which bring additional complexity such as [pagination](#) when the data sets are too large to hold easily in memory.

The term *representational state transfer* was introduced and defined in 2000 by [Roy Fielding](#) in his doctoral dissertation.^{[2][3]} Fielding's dissertation explained the REST principles that were known as the "HTTP object model" beginning in 1994, and were used in designing the [HTTP](#) 1.1 and [Uniform Resource Identifiers](#) (URI) standards.^{[4][5][6]} The term is intended to evoke an image of how a well-designed Web application behaves: it is a network of Web resources (a virtual state-machine) where the user progresses through the application by selecting links, such as `/user/tom`, and operations such as GET or DELETE (state transitions), resulting in the next resource (representing the next state of the application) being transferred to the user for their use.

4.3.2. User Interface

CRUD is also relevant at the user interface level of most applications. For example, in [address book](#) software, the basic storage unit is an individual *contact entry*. As a bare minimum, the software must allow the user to

- Create or add new entries
- Read, retrieve, search, or view existing entries
- Update or edit existing entries
- Delete/deactivate/remove existing entries

Without at least these four operations, the software cannot be considered complete. Because these operations are so fundamental, they are often documented and described under one comprehensive heading, such as "contact management", "content management" or "contact maintenance" (or "document management" in general, depending on the basic storage unit for the particular application).

test.js

```
1  const express = require('express')
2
3  const bodyParser = require('body-parser')
4
5
6  // create express app
7
8  const app = express()
9
10
11 // parse request of content-type - application/x-www-form-urlencoded
12
13 app.use(bodyParser.urlencoded({ extended : false}))
14
15
16 // parse request of content-type - application/json
17
18 app.use(bodyParser.json())
19
20
21 // define a simple route
22
23 app.get('/', (req, res) => {
24   res.json({ "Message": "welcome to the page" })
25 })
26
27
28 // listen for request
29
30 app.listen(3000, () => {
31   console.log("Server is listening on port number 3000")
32 })
```

First, We import express and body-parser modules. [Express](#), as you know, is a web framework that we'll be using for building the REST APIs, and [body-parser](#) is a module that parses the request (of various content types) and creates a `req.body` object that we can access in our routes.

Then, We create an express app, and add two `body-parser` middlewares using `express's app.use()` method. A `middleware` is a function that has access to the request and response objects. It can execute any code, transform the request object, or return a response.

Then, We define a simple GET route which returns a welcome message to the clients.

Finally, We listen on port 9999 for incoming connections.

4.4 Disk Storage

disk storage because store image store and json technology to store text file and using `multer` module for storing binary file and json for store information about user and its file upload information.

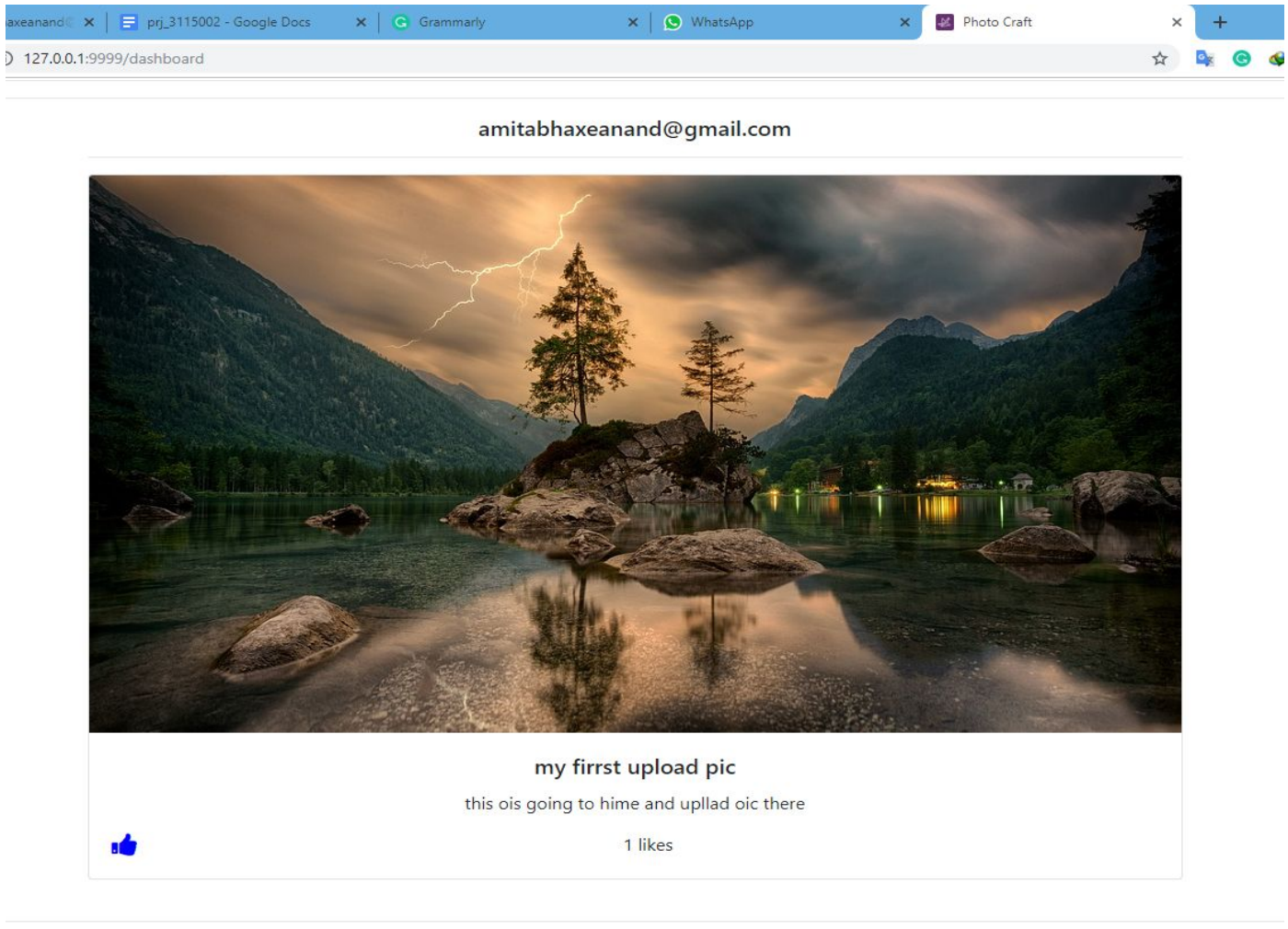
1. **Disk Directory** : disk directory for making user account folder and store its information like email folder 'photocraft@gmail.com' folder for storage.
2. **Multer** : `multer` store the binary file it store from client side to server side it is best to store client side and server

Disk Storage Description : now it only disk storage system no database attach currently , in future i will attach database like **mongodb or mysql** database because it is minor project for college and future with more guidance with our HOD. now it is done currently.


CHAPTER – 5

SNAPSHOTS


1. User Home page



2. User Login page

[Ranks](#) [Dashboard](#) [sign up](#) [sign in](#)


Store | Share | Innovate



☐ Remember me [Forgot password?](#)

[Sign In](#)

Not a member? [Sign Up](#)



Features
Cool stuff

Resources
Resource

Resources
Business

About
Team

3. User Dashboard

127.0.0.1:5000/dashboard

☆ 🔥 Inc

Ranks

Home

Card Title

Card Title

Card description

Enter detail of your photo

Upload photo


Upload

Browse

Upload

Progress Bar With Label

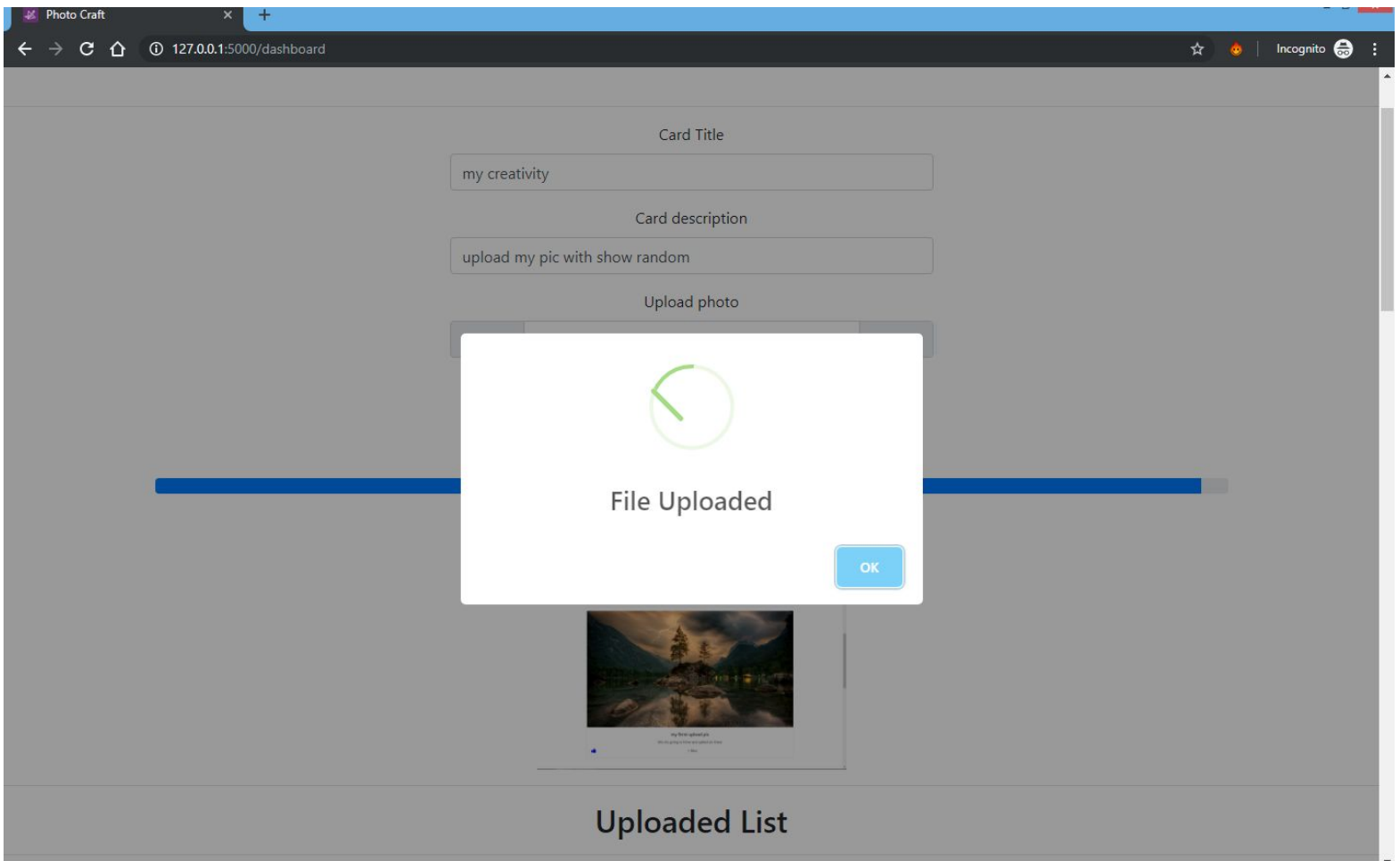
image preview

Image preview...

Uploaded List


amitaxeanand@gmail.com

4. User File Upload



4. User file uploaded files


127.0.0.1:5000/dashboard




okay

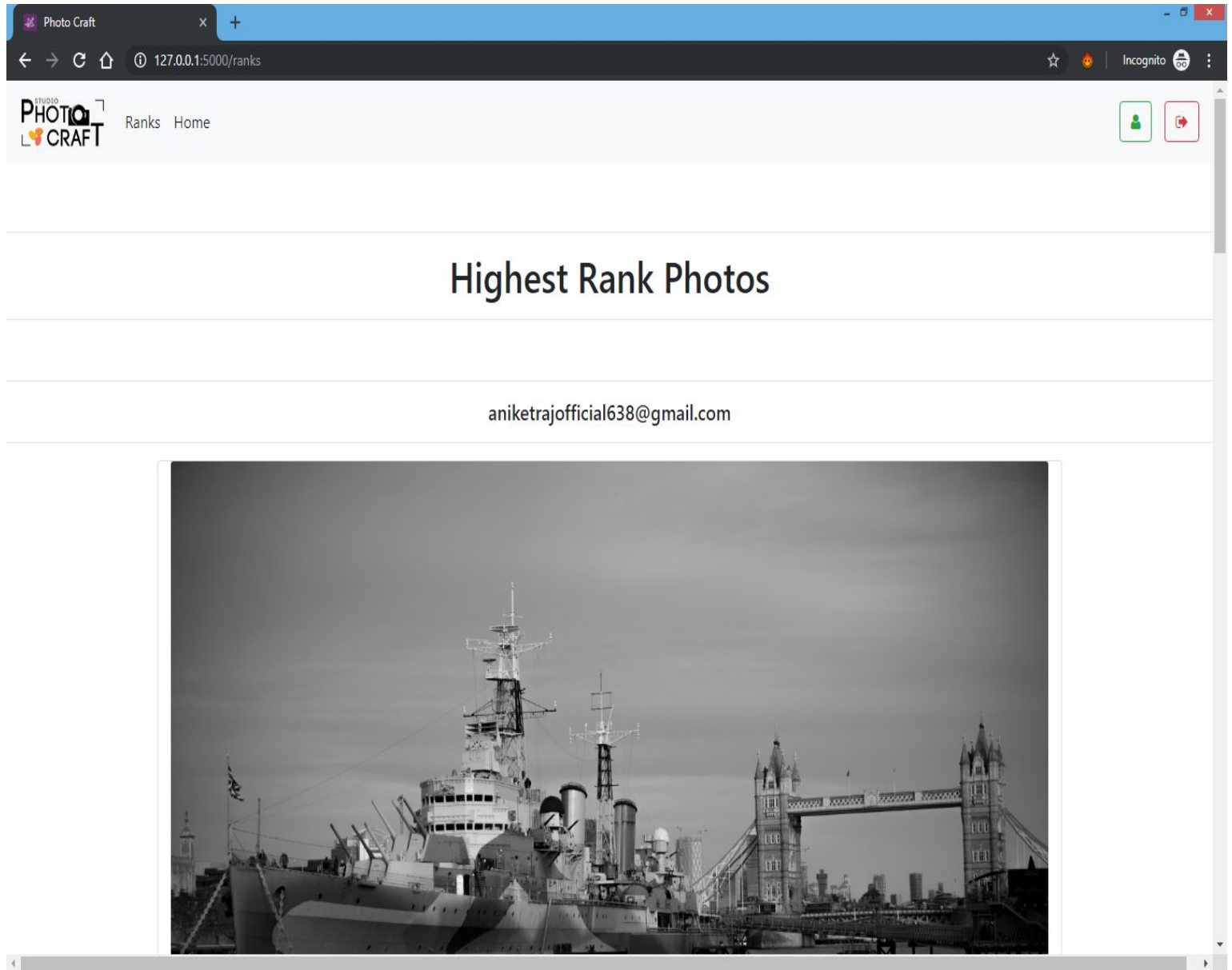
for best photo

1 likes

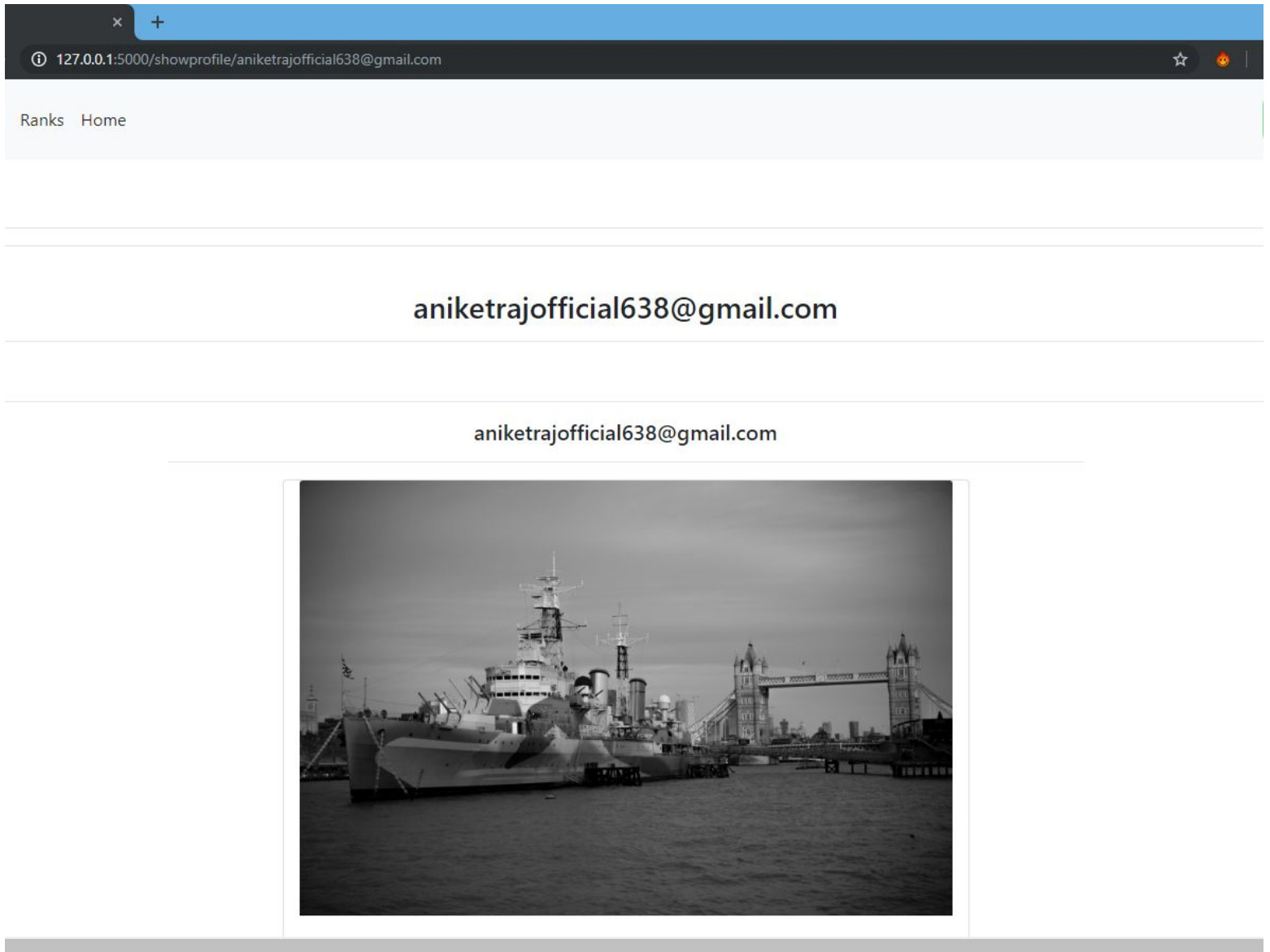




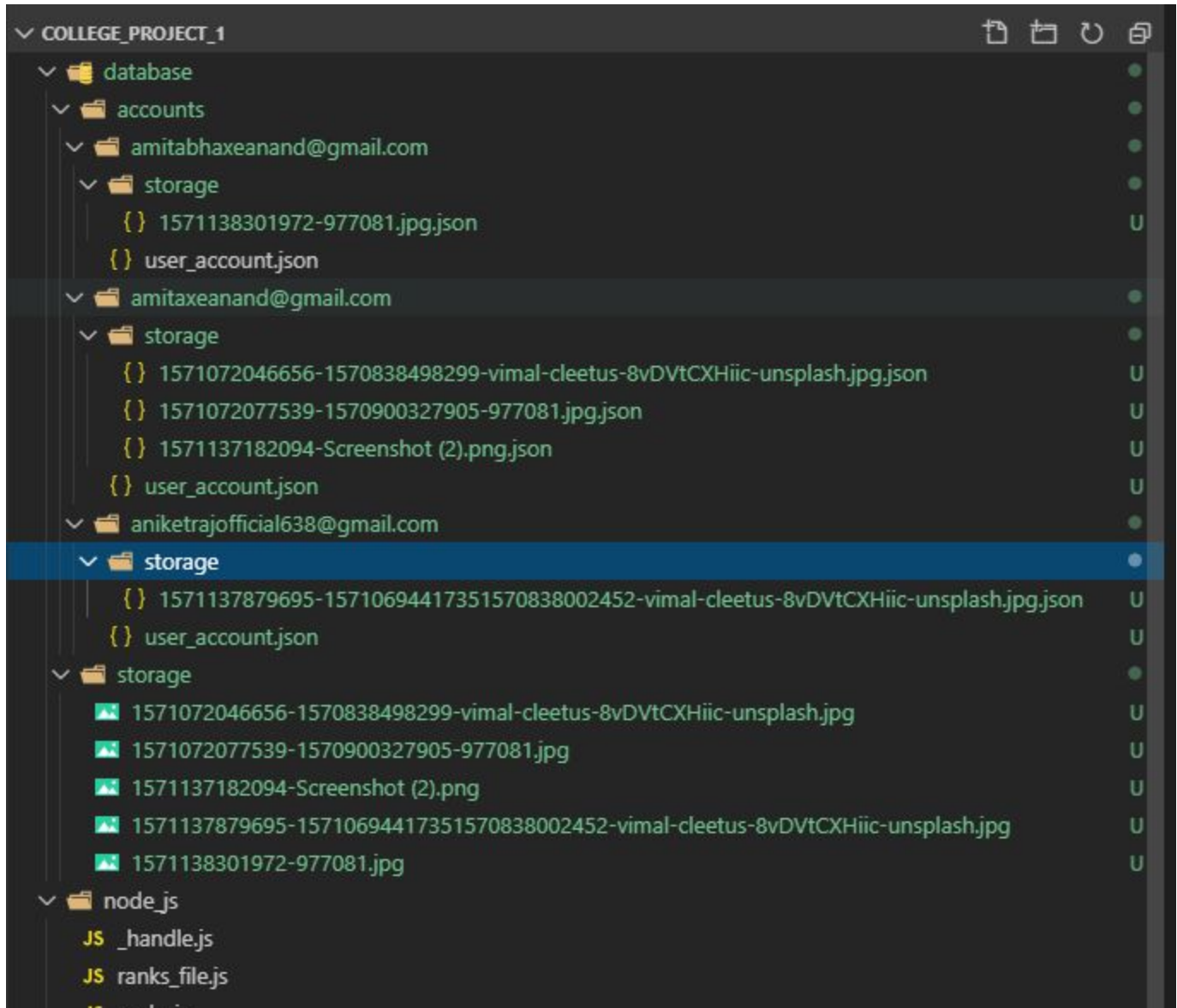
5. Ranks page to show all user image rank



6. single user anyone can access



7. Database Folder Structure



CHAPTER – 8

FUTURE SCOPE AND CONCLUSION

8.1 FUTURE SCOPE

Image is built right into our Drive, so you can work together in real time on documents,. You can share your stuff with others, add and reply to comments and even make edits on the go.

Timeline feature will come soon where every user can see the post, comment, like, dislike their post.

Search by the filter will also add soon where any user search the post by the author, by the special filter etc.

Features includes in next major version:

- 1.**Enhanced and more optimized file system for storage. Currently we are using array buffer but in next major version we will try to update on BST trees.
- 2.**To add cache based mechanism to minimize the overload and redundant fetching of the files and directory.
- 3.**Converting the front end into single page application to provide more native and progressive experience in the application.
- 4.**Real time editing of the files on the go not only for the text files but for all text based editing.
- 5.**Video player and Audio player to stream through others origins.

8.2 CONCLUSION

Node.js based website is a reliable web application which saves time and money as well as time complexity. This must be implemented in most of the information technology industries. It is friendly software or web application, through which user can easily interact and easy to manage their file and share it. This turns out to a great usage.

The system is capable of maintain details of various users and developers and storing all the day to day records data such as user file in the form of audio, video, images, documents etc. The central concept of the application is to allow the user and developers to use it virtually using internet and allow another user to see their current active updates.

“CR-I-SH” is your personal stash of files and folders that follows you wherever you go - it lets you keep everything and share anything. With this, you can access files, folders, and documents (such as Docs, Sheets, and Slides) from a web browser.

As free and decent cloud storage and service, “**PHOTO CRAFT**” enables you to create, share and keep all your stuff in one place. With this, you can now access your files, even the big ones, from wherever you are. **Share** them with whomever you want, and **edit** them together in **real time**.

References

Books:

1. Professional Node js, By Pedro Teixeira
2. Express In Action, By Evan. M. Hahn

Website:

1. <https://www.expressjs.com>
2. <https://www.nodejs.org>
3. <https://docs.mongodb.com>