Vectrofy

Submitted by

1. Falgun Sorathiya
2. Om Patel
3. Nishith Mehta
4. Naman Umraniya
5. Jaydeep Solanki

Department Name: Bachelor of Computer Application (BCA)

Institute Name: Silver Oak College of Computer Application (SOCCA)

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

INDEX

Abstract

Vectrofy is a cutting-edge image conversion tool built to transform raster images, such as PNG or JPG files, into infinitely scalable vector graphics (SVG). Unlike raster images, which are composed of pixels and suffer quality loss when resized, vector images consist of mathematically defined shapes, lines, and curves. This enables Vectrofy to produce graphics that can be resized to any dimension, from small icons to large billboards, without losing sharpness or clarity. The resulting SVG files are versatile and maintain their high quality, making them ideal for a wide range of applications.

At the heart of Vectrofy's technology is its ability to analyze colors, shapes, and intricate details within a raster image. Using advanced algorithms, Vectrofy detects the boundaries and structures of various elements in the image and converts them into mathematical representations, such as Bézier curves, paths, and polygons. This approach allows the software to break the image into distinct segments based on colors or shapes, accurately recreating it as a vector graphic. The conversion process preserves the essence of the original image while offering the benefits of scalability and editability.

In conclusion, Vectrofy provides an essential solution for modern design and development by delivering a high-precision tool for raster-to-vector conversion. It empowers users to scale their visuals infinitely, retain full control over design elements, and make edits seamlessly—all while maintaining the integrity of the original image. As the demand for scalable, editable, and efficient graphics grows across industries, Vectrofy stands out as a powerful asset for professionals seeking to streamline and optimize their design workflows.

Introduction

Vectrofy is a web-based application that addresses the growing need for scalable, high-quality graphics across various platforms. Traditional raster images, like PNG and JPG files, are often unsuitable for large-scale applications because they lose quality when resized. To overcome this limitation, Vectrofy converts these raster images into vector graphics (SVG), which are resolution-independent and maintain their sharpness regardless of scaling. By using Vectrofy, users can store and manage their images on the cloud, making it accessible from anywhere and ideal for collaborative workflows.

The core technology behind Vectrofy analyzes the colours, shapes, and fine details of an image to generate a vector file that is composed of mathematical paths and curves. This allows the image to be infinitely scalable without any degradation in quality, making it particularly valuable for graphic designers, web developers, and digital artists who require high-quality visuals across different sizes and mediums. The web application's cloud storage feature also enables users to easily upload, convert, and manage their design files online.

One of Vectrofy's key advantages is the versatility and flexibility it offers. The resulting SVG files are not only scalable but also highly editable. Users can adjust individual elements, such as shapes and colours, without affecting the overall image quality. This is essential for professionals who often need to modify designs to fit specific requirements, such as creating a logo that needs to be resized for both small social media icons and large marketing banners. Vectrofy ensures that users have full control over their design assets.

The cloud storage feature is a major advantage, allowing users to securely store their images and access them from any device. This supports collaboration and remote access, which are increasingly important in modern work environments. By centralizing image storage in the cloud, Vectrofy enhances workflow efficiency and reduces the

need for local storage, freeing up space on users' devices and providing the flexibility to work on projects from anywhere.

In conclusion, Vectrofy not only offers an intuitive and powerful solution for converting raster images into scalable vector graphics but also provides cloud storage for easy access and management of image files. Its combination of high-quality vector conversion, full editability, and cloud-based image storage makes it an invaluable tool for professionals looking to maintain quality and flexibility in their designs across various platforms and use cases.

Objective(s)

1. **Raster to Vector Conversion:** To develop a reliable and efficient web application that converts raster images (PNG, JPG) into scalable vector graphics (SVG), enabling users to maintain high quality regardless of size or resolution.

2. **User-Friendly Interface:** To create a colorful and modern user interface that enhances user experience, making the image conversion process intuitive and accessible for both novice and experienced users.

3. **Image Analysis and Processing:** To implement sophisticated algorithms that accurately analyze the colors, shapes, and details of raster images, ensuring precise vectorization and the retention of the original image's integrity during conversion.

4. **Editability and Customization:** To provide users with comprehensive editing tools that allow them to modify and customize vector graphics easily, including options to adjust colors, shapes, and paths without compromising quality.

5. **Cloud Storage Integration:** To incorporate a cloud storage feature that allows users to securely save and manage their images online, facilitating easy access, collaboration, and organization of design files from any device.

6. **Performance Optimization:** To ensure that the application performs efficiently, even with complex images, by optimizing

the conversion process and minimizing loading times, thereby enhancing overall user satisfaction.

7. **Scalability and Future Expansion:** To design Vectrofy with scalability in mind, allowing for future enhancements and features, such as AI-driven suggestions for image improvement and additional export formats, to keep pace with evolving user needs and industry trends.

Methodology

- **Requirement Analysis:**

In developing Vectrofy, a comprehensive requirement analysis was conducted to ensure the application meets the diverse needs of its target users while utilizing modern technologies effectively. The analysis encompassed both functional and non-functional requirements, covering the entire architecture of the application, from the user interface to backend processing and security features.

Functional Requirements:

1. **Image Conversion Functionality:** The core feature of Vectrofy is its ability to convert raster images (PNG, JPG) into scalable vector graphics (SVG). This necessitates implementing robust algorithms capable of analyzing image properties and accurately transforming them into vector format, ensuring high fidelity in the conversion process.

2. **User Interface Design:** A colorful and modern user interface (UI) is essential for enhancing user experience. The UI should facilitate easy image uploads, display conversion progress, and allow users to download the resulting SVG files. Additionally, the design must be responsive to ensure compatibility across various devices and screen sizes.

3. **User Authentication:** To enhance security and personalization, Vectrofy will include user authentication functionalities such as login and signup. This feature will allow users to create accounts, securely log in, and access their stored images in the cloud. User authentication will also facilitate personalized experiences, allowing users to save their preferences and design history.

4. **Machine Learning Integration:** Machine learning techniques implemented in Python will improve the accuracy and efficiency of the image conversion process. These algorithms will analyze the intricacies of the images and optimize vectorization. FastAPI will serve as the communication layer between the front-end and the backend, ensuring smooth integration.

5. **Cloud Storage:** The application will offer a reliable cloud storage solution for securely saving and managing images. Users should be able to easily access their saved files and organize them for efficient retrieval, enhancing their overall experience.

Non-Functional Requirements:

1. **Performance:** Vectrofy should be optimized for speed and responsiveness, ensuring that the conversion process is quick—even for complex images. The overall application must maintain minimal loading times to enhance user satisfaction.

2. **Scalability:** The application should be designed to accommodate an increasing number of users and images. This involves ensuring both the front-end and backend can scale effectively to handle high traffic without degradation in performance.

3. **Security:** Robust security measures are essential, especially given that users will be uploading and storing images. User authentication should include secure login processes, data encryption, and protection against unauthorized access, ensuring user data is safe.

4. **Maintainability:** The codebase should adhere to best practices to ensure easy maintenance and future enhancements. Clear documentation, modular design, and adherence to coding standards will facilitate ongoing development and updates.

5. **Usability:** The application must be user-friendly, featuring intuitive navigation and clear instructions. Incorporating user feedback mechanisms will allow for gathering insights that help refine the application and ensure it effectively meets user needs.

By addressing these functional and non-functional requirements, the Vectrofy project aims to deliver a robust web application that leverages the strengths of the MERN stack, machine learning capabilities, and modern design principles. The addition of user authentication will enhance security and personalization, making Vectrofy an indispensable tool for users in image conversion and editing.

- **System Design:**

The system design of Vectrofy outlines the architecture, components, and interactions necessary to support its functionality as a web-based application for converting raster images to scalable vector graphics (SVG). This section details the overall architecture, database design, front-end and back-end components, and integration of machine learning algorithms.

1. Architecture Overview

Vectrofy follows a microservices architecture that separates the application into distinct components, each responsible for specific functionalities. This approach enhances modularity, scalability, and maintainability. The key components include:

- **Front-End**: Developed using the MERN stack (MongoDB, Express.js, React, Node.js), the front-end handles user interactions, provides a colorful and modern user interface, and communicates with the back-end services.
- **Back-End**: Comprising Node.js and Express.js, the back-end manages API requests, user authentication, and image processing tasks. It serves as the intermediary between the front-end and the machine learning services.
- **Machine Learning Service**: Implemented in Python, this component utilizes sophisticated algorithms to analyze and convert images to vector format. FastAPI is employed to facilitate seamless integration with the Node.js backend.
- **Cloud Storage**: User images and converted SVG files are securely stored in a cloud storage solution, allowing for easy access and management.

2. Database Design

The database design for Vectrofy is based on a NoSQL approach using MongoDB. The key collections include:

- **Users Collection**: Stores user information such as username, email, hashed passwords, and preferences. This collection supports user authentication and allows users to maintain their accounts securely.
- **Images Collection**: Contains metadata for uploaded images, including the original file format, conversion status, user ID (to link to the user who uploaded the image), and a reference to the stored SVG file in the cloud.
- **Edit History Collection**: Tracks the editing changes made by users to their vector files, enabling them to revert to previous versions if needed.

3. Front-End Components

The front-end of Vectrofy is built using React, which allows for dynamic user interactions and efficient rendering of components. Key front-end features include:

- **User Authentication**: Implementing login and signup forms, along with session management to maintain user sessions securely.
- **Image Upload and Conversion**: A user-friendly interface for uploading raster images, monitoring conversion progress, and downloading the resulting SVG files.
- **Responsive Design**: A layout that adapts to various screen sizes and devices, ensuring a consistent user experience.

4. Back-End Components

The back-end is responsible for handling API requests, user management, and image processing. Key back-end functionalities include:

- **RESTful API**: An API built using Express.js that manages user authentication, image uploads, and retrieval of converted SVG files.
- **User Authentication**: Implementation of login/signup for user authentication and session management.
- **Image Processing**: Handling of image upload requests, triggering the machine learning service for conversion, and managing the storage of the resulting SVG files.

5. Machine Learning Integration

The machine learning service is a critical component that enhances the image conversion process. Key aspects include:

- **Algorithm Development**: Algorithms are developed in Python to analyze raster images, detecting colors, shapes, and edges to generate accurate vector representations.
- **FastAPI Integration**: FastAPI serves as a bridge between the machine learning service and the Node.js backend, allowing for efficient communication and data exchange during the conversion process.

6. Cloud Storage

Vectrofy utilizes cloud storage solutions (e.g., AWS S3, Google Cloud Storage) for storing user images and converted SVG files. Key features include:

- **Secure File Storage**: Images are stored securely, ensuring data integrity and protection against unauthorized access.
- **Scalable Storage Solutions**: The cloud storage can handle an increasing volume of user data, allowing for future growth without performance degradation.

▪ Software Development:

The system architecture of Vectrofy is designed to ensure scalability, modularity, and efficient communication between its components. It is divided into a frontend and backend, developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js), with an additional Python-based microservice handling the complex task of image processing. This separation of concerns allows each component to operate independently while communicating seamlessly with the others. The frontend, built using React, serves as the user-facing interface, providing users with the ability to upload images, view

conversion progress, and download the resulting vector files. It also features a colorful, modern UI, which is both responsive and user-friendly across devices.

The React frontend interacts directly with the backend via RESTful APIs. The backend, developed using Node.js and Express, manages API requests, authentication, and session handling. It securely processes user uploads and sends them to the image processing microservice for conversion. MongoDB Atlas, a cloud-based NoSQL database, is used to store user data, such as login credentials, as well as image metadata, enabling users to manage their image histories and access their cloud-stored files at any time. The image stored in cloud is in string format as Vectrofy converts images into base64 format, which helps to reduce the storage space in cloud.

A key feature of the frontend is the range bar, which allows users to manipulate the threshold value for image conversion. This adjustable slider gives users more control over how their raster image is vectorized, allowing for finer tuning of the conversion process. By adjusting the threshold, users can influence how shapes are detected, enabling them to achieve the level of detail or abstraction they desire in the final vector output. This interactive component enhances the customization and precision of the image conversion experience.

- **Algorithm Development:**

The algorithm development for Vectrofy's image conversion functionality was a critical aspect of the project, as it directly impacts the accuracy and quality of the SVG output. The core of the image conversion algorithm was developed using Python, leveraging powerful libraries such as **Pillow** and **svgwrite**. Pillow, a widely-used image processing library, enables the application to read and manipulate raster images in formats like PNG and JPG. It provides tools to analyze pixel data, detect edges, and distinguish between different regions of color. These capabilities are essential for breaking

down an image into its basic components before conversion into vector format.

Once the image has been analyzed and segmented, **svgwrite** is used to generate the corresponding SVG file. This library allows the algorithm to define the shapes, lines, and curves mathematically, preserving the original image's structure while ensuring infinite scalability. The algorithm maps image segments—such as colors, shapes, and edges—into vector paths, ensuring that the final SVG file is precise and visually faithful to the original raster image. Additionally, the vector format enables users to scale the image without any degradation in quality, making it suitable for a wide range of design applications.

- **Integration and Testing:**

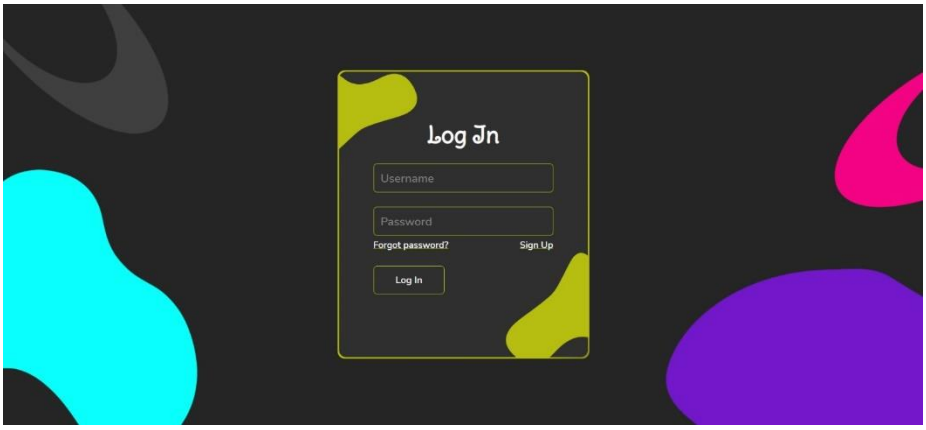
  The **integration phase** of Vectrofy involved linking the various components of the system, ensuring that the front-end, back-end, machine learning service, and cloud storage work seamlessly together. A critical part of this process was integrating the Python-based image conversion service with the MERN stack application. This was achieved using a Flask API, which acted as the communication bridge between the React frontend and the Python algorithms responsible for converting raster images into SVG format. The integration focused on ensuring smooth data flow, secure communication, and efficient image processing, all while maintaining a responsive user experience.

  Flask API Integration: The Flask API was developed to receive image upload requests from the React-based frontend. Upon image upload, the front-end sends the image data, along with any selected threshold value, to the Flask API. The Flask service processes the image using the Python libraries Pillow and svgwrite, converting the raster image (PNG or JPG) into a scalable vector graphic (SVG). After conversion, the API sends the SVG back to the frontend, where the user can view or download the result. To ensure real-time responsiveness, the API handles multiple image

processing requests concurrently while maintaining low latency, enhancing the overall user experience. The communication between the React frontend and Flask API was secured using HTTPS to protect the integrity of the image data being transferred.

The **testing phase** of Vectrofy was conducted manually by team members to ensure that all features functioned as intended and that the integration between components was robust. Testing was carried out in several stages, focusing on the core functionalities, performance under different conditions, and user experience.

- **Photos:**



1.1 Authentication Page

1.2 Main Page

Outcomes

Vectrofy successfully achieved its primary objectives of delivering a robust and efficient cloud-based solution for converting raster images into vector graphics. The system enables users to upload raster images in formats like PNG or JPG, which are then processed and converted into SVG files that maintain scalability without any loss in quality. By offering this service, Vectrofy addresses the growing need for scalable, high-precision images in design, development, and various industries. The user-friendly interface allows for easy uploads and customization of conversion parameters, ensuring that users have control over the final vector output. Additionally, the platform enables users to access their previously uploaded files, offering a convenient and organized cloud storage system for managing multiple image projects.

References

Udemy:

- MERN Course: https://www.udemy.com/course/react-nodejs-express-mongodb-the-mern-fullstack-guide/

YouTube:

- Redux Tutorial: https://youtube.com/watch?v=1i04-A7kfFI&t=3245s

- Image Upload in Cloud: https://www.youtube.com/watch?v=YH63fnqG7F0&t=1s

- Raster Image to Vector Conversion: https://youtu.be/i_9tAee-hYo?si=snPzcD_ujSKaS7TU

Teaching Websites:

- Flask Python and React Integration: https://www.geeksforgeeks.org/flask-tutorial/