

# **Software Design Description for Virtual Guiding System**

Version 1.0

Group No:15

**Prepared by**

**MIKHEL V KUTTICKAL (41)**

**MEGHNA A G (38)**

**PARVATHY J (50)**

**DHRISYA T R (77)**

**03-04-2023**

# **Table of Contents**

## **1. Introduction**

- 1.1 Purpose
- 1.2 Scope
- 1.3 Document Conventions
- 1.4 References

## **2. System Overview**

- 2.1 System Architecture
- 2.2 System Design
- 2.3 System Components

## **3. Design Considerations**

- 3.1 Assumptions and Dependencies
- 3.2 Constraints
- 3.3 Risks and Mitigations

## **4. Design Details**

- 4.1 System Architecture Diagram
- 4.2 Chatbot Design
- 4.3 Technology Stack

## **5. Testing**

# **1. Introduction**

## **1.1 Purpose**

The purpose of this Software Design Document (SDD) is to provide a detailed design specification for the Virtual Guiding System for College Students. This document outlines the system architecture, components, design details, and considerations that will guide the development of the system.

## **1.2 Scope**

The Virtual Guiding System is a conversational chatbot system designed to help college students navigate their campus and access information about their courses, schedules, and campus resources through WhatsApp. The system will provide guidance, information, and updates via a conversational interface that can be accessed from any device with WhatsApp installed. The scope of this SDD is to provide a detailed design specification for the system.

## **1.3 Document Conventions**

The following conventions will be used throughout this document:

Bold text will be used for section headings and subheadings

Italic text will be used for terms and definitions

"Quotes" will be used for user input or system output

Bullet points will be used for lists

UML diagrams may be used to illustrate system architecture and design

## **1.4 References**

IEEE Std 1016-2009, IEEE Standard for Information Technology -- Systems Design -- Software Design Descriptions

<https://ieeexplore.ieee.org/document/720574>

<https://developers.facebook.com/docs/whatsapp/cloud-api/>

<https://nodejs.org/en/download/>

<https://webhook.site/#!/7d284e30-ec75-4cdc-baac-47aeb3fdbd72>

Software development life cycle methodologies, Agile.

# **2. System Overview**

## **2.1 System Architecture**

The Virtual Guiding System will be designed as a chatbot system that integrates with WhatsApp Business API. The system architecture will be composed of several components, including the WhatsApp Business API, WhatsApp cloud, webhook, and a web server to deploy.

## **2.2 System Design**

The system design will be based on a modular approach, with each module responsible for a specific set of functionalities. The system will be designed to handle user requests and queries, and retrieve user data, and provide relevant responses to users. The design will also include a user interface that is simple, intuitive, and easy to use.

## **2.3 System Components**

The Virtual Guiding System will consist of the following components:

*WhatsApp Business API:* allows developers to integrate WhatsApp messaging functionality into their own applications or services.

*Webhook:* Automated event notification between applications in real- time.

*WhatsApp Cloud:* Provides whatsapp API

*Web Server:* For deploying (webhook endpoint)

## **3. Design Considerations**

### **3.1 Assumptions and Dependencies**

The following assumptions and dependencies have been identified for the Virtual Guiding System:

The system will be hosted on a cloud-based server to ensure scalability and accessibility.

The system will depend on WhatsApp Business API for user interactions and messaging.

The system will use python programming language for creating chatbot.

The system will require access to campus data and resources, such as course schedules and maps.

### **3.2 Constraints**

The following constraints have been identified for the Virtual Guiding System:

The system must adhere to WhatsApp Business API guidelines and best practices.

The system must ensure data privacy and security for user information.

The system must ensure scalability and performance for a large number of users.

The system must ensure accuracy and relevancy of responses to user queries.

The system must be designed for simplicity and ease of use.

### **3.3 Risks and Mitigations**

The following risks have been identified for the Virtual Guiding System:

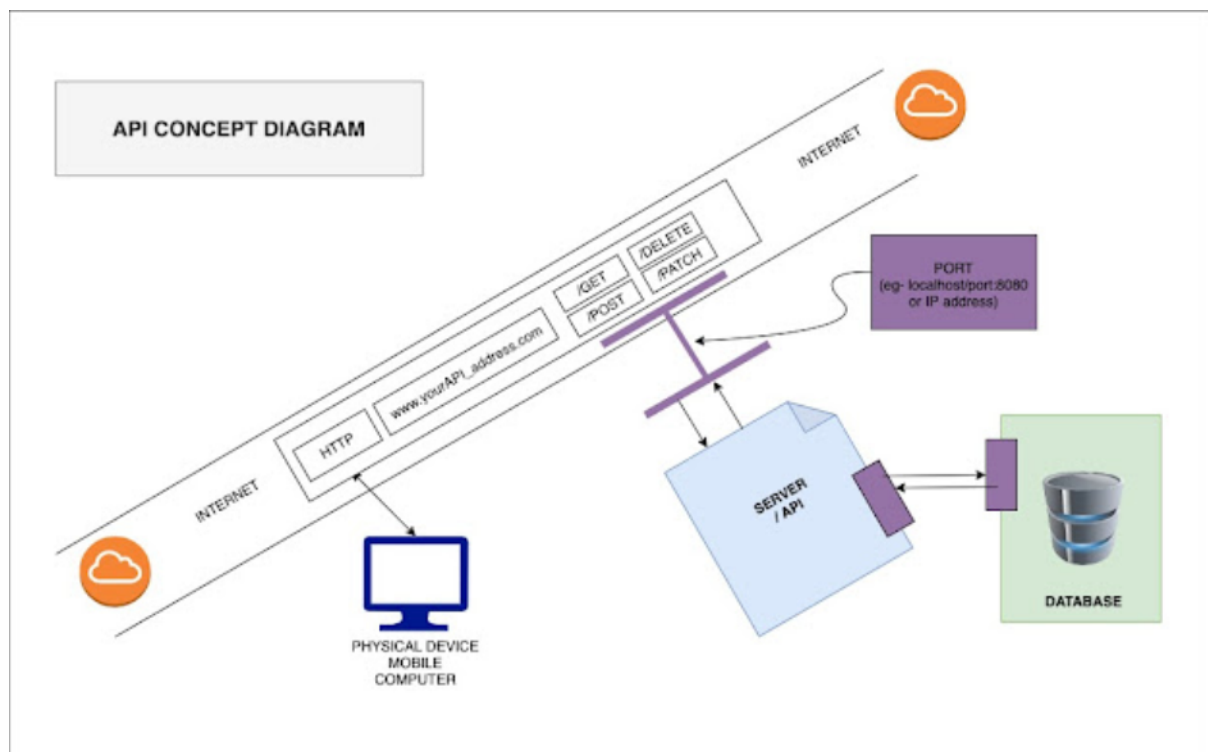
**Technical Risks:** There may be technical issues related to server downtime, network connectivity, or third-party dependencies. To mitigate these risks, the system will be designed with fault tolerance and redundancy in mind, and regular backups and monitoring will be conducted.

**Data Security Risks:** There may be risks related to data privacy and security, such as data breaches or unauthorised access. To mitigate these risks, the system will be designed with secure protocols and encryption, and regular security audits and vulnerability testing will be conducted.

**User Acceptance Risks:** There may be risks related to user acceptance and adoption of the system. To mitigate these risks, the system will be designed with user experience and feedback in mind, and regular user testing and feedback will be conducted.

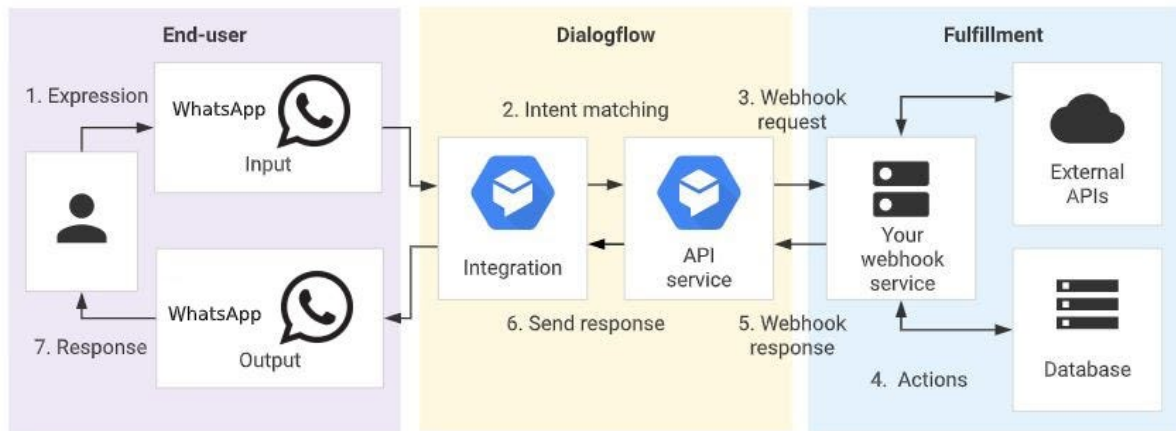
## 4. Design Details

### 4.1 System Architecture Diagram:



The user interacts with the virtual guide through the WhatsApp interface, which sends the requests to the application server. The application server processes the requests and retrieves information from the database, if necessary. The database stores information related to college events, activities, academic and career advice, and mentoring resources. The application server then sends the response back to the WhatsApp interface, which displays it to the user.

To ensure the system's reliability and scalability, the application server can be designed as a distributed system with load balancing and failover mechanisms. The database can be replicated and synchronised to ensure data consistency and availability. Additionally, the system can be integrated with other college systems, such as the student information system, to provide more personalised and relevant guidance to the students.



## 4.2 Chatbot Design

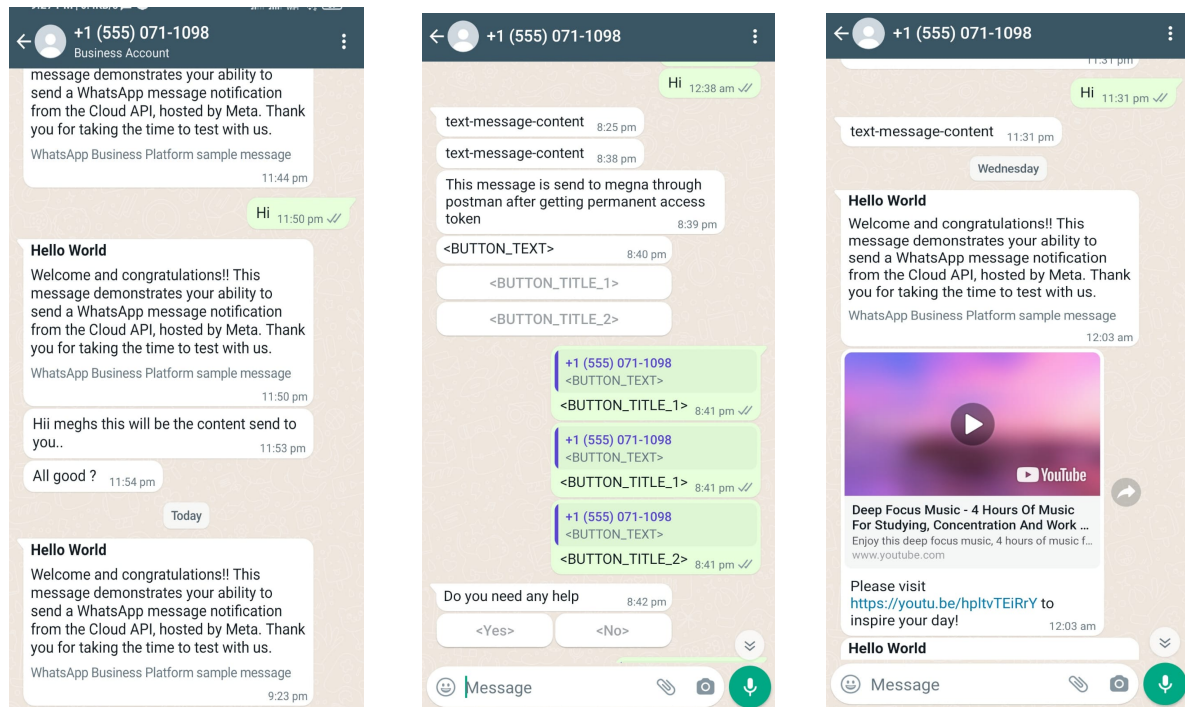
The Virtual Guiding System will be designed as a conversational chatbot that can provide guidance, information, and updates to college students. The chatbot will have the following features:

**Welcome Message:** The chatbot will provide a welcome message to new users and ask for their queries.

**Information:** The chatbot will provide information about activities in the college, the schedules, and related information.

**Campus Navigation:** The chatbot will provide directions and maps to campus locations, such as buildings, offices, labs, halls and classrooms.

The chatbot also provides guidance for beginners for skill development and also addresses their queries if any.



### 4.3 Technology Stack

- **Railway app:** Railway deployment refers to the process of deploying and hosting an application on the Railway platform, which is a cloud-based platform for deploying web applications. The process involves several steps, such as creating an account on the platform, connecting a code repository, configuring the deployment settings, and deploying the application.
- **Postman:** Postman is a popular API client tool used for testing, designing, and documenting APIs. It allows developers to easily send HTTP requests to APIs and view the responses. Postman supports various HTTP methods such as GET, POST, PUT, DELETE, etc. and also supports authentication mechanisms like OAuth, Basic Auth, and API keys. Additionally, Postman provides features such as automated testing, API documentation, and collaboration among team members. It is available as a desktop application for Windows, macOS, and Linux, as well as a Chrome extension.
- **WhatsApp API server** is a server that allows developers to integrate WhatsApp messaging functionality into their own applications or services. It acts as an intermediary between the application and WhatsApp, providing a way to send and receive messages, as well as manage contacts .
- **Webhook:** A webhook is a way for an application to provide other applications with real-time information. It is essentially a user-defined HTTP callback. When a certain event occurs, the application that registered the webhook is notified via an HTTP POST request with a payload of relevant data. This allows the application to receive and process data in real-time without having to continuously poll an API for updates. Webhooks are commonly used for various purposes, such as integrating with third-party services, receiving notifications, and automating workflows.
- **GitHub** is a web-based platform that provides a hosting service for version control using Git. It allows individuals and teams to collaborate on software development projects and maintain

code repositories. Users can create and contribute to open-source projects, track changes to code over time, and collaborate with others using features like pull requests and code reviews. GitHub also provides a range of other features, such as issue tracking, project management tools, and integration with other development tools and services.

- Node.js is an open-source, cross-platform JavaScript runtime environment that enables developers to build server-side applications using JavaScript. It uses an event-driven, non-blocking I/O model, which makes it lightweight and efficient for building scalable and high-performance applications. Node.js is built on top of the Google V8 JavaScript engine and provides a rich library of modules and packages that developers can use to build web servers, command-line tools, real-time applications, and more.

## 5. Testing

**Postman:** Postman is a popular API client tool used for testing, designing, and documenting APIs. It allows developers to easily send HTTP requests to APIs and view the responses.

Testing is a critical part of the API development process. You can create a collection that contains your API tests and link it to your API.

### Adding API tests

You can connect a test collection (a collection containing API tests) to an API you've defined in the Postman API Builder.

To add a test collection to an API, do the following:

1. Select APIs in the sidebar and select an API.
2. Select Test and Automation.
3. Next to Collections, select + and select an option:

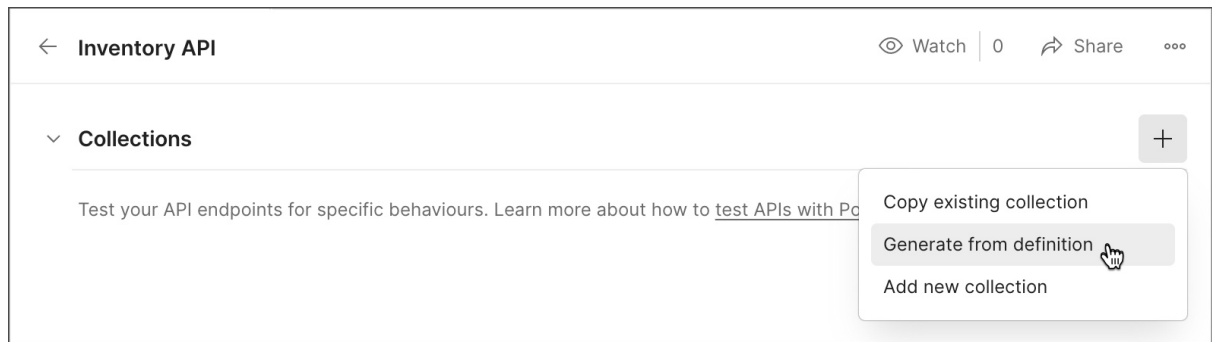
Add new collection - This option creates a new empty collection in the API.

You can add your tests to the Tests tab.

Copy existing collection - Select an available collection from the list. A copy of the collection is added to the API.

Generate from definition - Change any settings to customise the new collection and select Generate Collection.




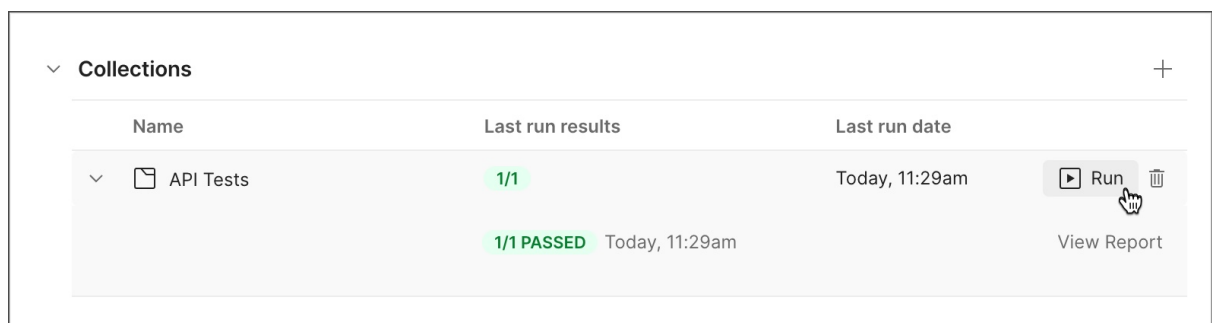


## Running API tests

After adding a test collection, you can run the collection to test your API and view test results.

To run a test collection for an API, do the following:

1. Select APIs in the sidebar and select an API.
2. Select Test and Automation.
3. Under Collections, select  Run next to a test collection.
4. Select any configuration options for the collection run, then select Run API Tests.
5. To view detailed test results, expand the collection and select View Report next to a test run.



To remove a test collection from an API, select the delete icon  next to the collection.

## Adding CI integration

Postman integrates with some of the most widely used continuous integration and continuous delivery (CI/CD) tools. After you set up CI integration for your API, you can view the status

of builds or start a new build, all from within Postman. You can also run API tests created in Postman as part of your CI pipeline.

To set up a CI integration for an API, do the following:

1. Select APIs in the sidebar and select an API.
2. Select Test and Automation.
3. Under Automate, select the CI integration you want to add.

