**Abstract**

Automation is the technology by which a process or procedure is performed with minimal human assistance.Automation or automatic control is the use of various [controlsystems](https://en.wikipedia.org/wiki/Control_system) for operating equipment such as machinery, processes in factories, boilers and heat treating ovens, switching on telephone networks, steering and stabilization of ships, aircraft and other applications and vehicles with minimal or reduced human intervention.Automation covers applications ranging from a household [thermostat](https://en.wikipedia.org/wiki/Thermostat) controlling a boiler, to a large industrial control system with tens of thousands of input measurements and output control signals. In control complexity, it can range from simple on-off control to multi-variable high-level algorithms.In the simplest type of an automatic [control loop](https://en.wikipedia.org/wiki/Control_loop), a controller compares a measured value of a process with a desired set value, and processes the resulting error signal to change some input to the process, in such a way that the process stays at its set point despite disturbances. This closed-loop control is an application of negative feedback to a system. The mathematical basis of [control theory](https://en.wikipedia.org/wiki/Control_theory) was begun in the 18th century and advanced rapidly in the 20th.Automation has been achieved by various means including mechanical, [hydraulic](https://en.wikipedia.org/wiki/Hydraulics), [pneumatic](https://en.wikipedia.org/wiki/Pneumatics), electrical, electronic devices and [computers](https://en.wikipedia.org/wiki/Computer), usually in combination. Complicated systems, such as modern factories, [airplanes](https://en.wikipedia.org/wiki/Airplane) and [ships](https://en.wikipedia.org/wiki/Ship) typically use all these combined techniques. The benefit of automation includes labor savings, savings in [electricitycosts](https://en.wikipedia.org/wiki/Electricity), savings in material costs, and improvements to quality, accuracy, and precis

**1.INTRODUCTION**

**1.1 INTRODUCTION:**

Automation is the technology by which a process or procedure is performed with minimal human assistance. Automation or automatic control is the use of various control systems for operating equipment such as machinery, processes in factories, boilers and heat-treating ovens, switching on telephone networks, steering and stabilization of ships, aircraft and other applications and vehicles with minimal or reduced human intervention.

Automation covers applications ranging from a household thermostat controlling a boiler, to a large industrial control system with tens of thousands of input measurements and output control signals. In control complexity, it can range from simple on-off control to multi-variable high-level algorithms.

In the simplest type of an automatic control loop, a controller compares a measured value of a process with a desired set value, and processes the resulting error signal to change some input to the process, in such a way that the process stays at its set point despite disturbances. This closed-loop control is an application of negative feedback to a system. The mathematical basis of control theory was begun in the 18th century and advanced rapidly in the 20th.

Automation has been achieved by various means including mechanical, hydraulic, pneumatic, electrical, electronic devices and computers, usually in combination. Complicated systems, such as modern factories, airplanes and ships typically use all these combined techniques. The benefit of automation includes labor savings, savings in electricity costs, savings in material costs, and improvements to quality, accuracy, and precision.

The World Bank's World Development Report 2019 shows evidence that the new industries and jobs in the technology sector outweigh the economic effects of workers being displaced by automation.

Job losses and downward mobility blamed on Automation has been cited as one of many factors in the resurgence of nationalist and protectionist politics in the US, UK and France, among other countries since 2010s.

The term automation, inspired by the earlier word automatic (coming from automaton), was not widely used before 1947, when Ford established an automation department. It was during this time that industry was rapidly adopting feedback controllers, which were introduced in the 1930s.

Automation has a single purpose: To let machines perform repetitive, monotonous tasks. This frees up time for fleshy humans to focus on more important tasks that require the personal touch. The end result is a more efficient, cost-effective business and a more productive workforce.

**1.2 PROPOSED SYSTEM:**

Here we proposed Automation with Alexa, to get work or task done thought with Alexa in easiest manner.

**1.3Advantages of Proposed System:**

* Increased throughput or productivity.
* Improved quality or increased predictability of quality.
* Improved robustness (consistency), of processes or product.
* Increased consistency of output.
* Reduced direct human labor costs and expenses.
* Installation in operations reduces cycle time.
* Can complete tasks where a high degree of accuracy is required.
* Replaces human operators in tasks that involve hard physical or monotonous work (e.g., using one forklift with a single driver instead of a team of multiple workers to lift a heavy object)
* Reduces some occupational injuries (e.g., fewer strained backs from lifting heavy objects)
* Replaces humans in tasks done in dangerous environments (i.e. fire, space, volcanoes, nuclear facilities, underwater, etc.)
* Performs tasks that are beyond human capabilities of size, weight, speed, endurance, etc.
* Reduces operation time and work handling time significantly.
* Frees up workers to take on other roles.
* Provides higher level jobs in the development, deployment, maintenance and running of the automated processes.

**2. LITERATURE SURVEY**

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html) or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

**“SPANStore: Cost-effective Geo-replicated Storage Spanning Multiple Cloud Services,”**

Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha

By offering storage services in several geographically distributed data centers, cloud computing platforms enable applications to offer low latency access to user data. However, application developers are left to deal with the complexities associated with choosing the storage services at which any object is replicated and maintaining consistency across these replicas. In this paper, we present SPANStore, a key-value store that exports a unified view of storage services in geographically distributed data centers. To minimize an application provider’s cost, we combine three key principles. First, SPANStore spans multiple cloud providers to increase the geographical density of data centers and to minimize cost by exploiting pricing discrepancies across p+roviders. Second, by estimating application workload at the right granularity, SPANStore judiciously trades off greater geo-distributed replication necessary to satisfy latency goals with the higher storage and data propagation costs this entails in order to satisfy fault tolerance and consistency requirements. Finally, SPANStore minimizes the use of compute resources to implement tasks such as two-phase locking and data propagation, which are necessary to offer a global view of the storage services that it builds upon. Our evaluation of SPANStore shows that it can lower costs by over 10x in several scenarios, in comparison with alternative solutions that either use a single storage provider or replicate every object to every data center from which it is accessed.

**“Robust Data Sharing with Key-value Stores,”**

C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic

A key-value store (KVS) offers functions for storing and retrieving values associated with unique keys. KVSs have become widely used as shared storage solutions for Internet-scale distributed applications. We present a fault-tolerant wait-free efficient algorithm that emulates a multi-reader multi-writer register from a set of KVS replicas in an asynchronous environment. Our implementation serves an unbounded number of clients that use the storage. It tolerates crashes of a minority of the KVSs and crashes of any number of clients. We provide two variants of our algorithm: one implementing an atomic register and one implementing a regular register; the latter does not require read operations to store data at the underlying KVSs. We note that applying state-of-the-art reliable storage solutions to this scenario is either impossible or prohibitively inefficient.

**“Secret-sharing schemes: A survey,” in International Workshop on Coding and Cryptology (IWCC),**

A secret-sharing scheme is a method by which a dealer distributes shares to parties such that only authorized subsets of parties can reconstruct the secret. Secret-sharing schemes are an important tool in cryptography and they are used as a building box in many secure protocols, e.g., general protocol for multiparty computation, Byzantine agreement, threshold cryptography, access control, attribute-based encryption, and generalized oblivious transfer. In this survey, we describe the most important constructions of secretsharing schemes; in particular, we explain the connections between secretsharing schemes and monotone formulae and monotone span programs. We then discuss the main problem with known secret-sharing schemes – the large share size, which is exponential in the number of parties. We conjecture that this is unavoidable. We present the known lower bounds on the share size. These lower bounds are fairly weak and there is a big gap between the lower and upper bounds. For linear secret-sharing schemes, which is a class of schemes based on linear algebra that contains most known schemes, super-polynomial lower bounds on the share size are known. We describe the proofs of these lower bounds. We also present two results connecting secret-sharing schemes for a Hamiltonian access structure to the NP vs. coNP problem and to a major open problem in cryptography – constructing oblivious-transfer protocols from one-way functions.

**“DepSky: Dependable and Secure Storage in a Cloud-ofclouds,”**

A.Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa,

The increasing popularity of cloud storage services has lead companies that handle critical data to think about using these services for their storage needs. Medical record databases, power system historical information and financial data are some examples of critical data that could be moved to the cloud. However, the reliability and security of data stored in the cloud still remain major concerns. In this paper we present DEPSKY, a system that improves the availability, integrity and confidentiality of information stored in the cloud through the encryption, encoding and replication of the data on diverse clouds that form a cloud-of-clouds. We deployed our system using four commercial clouds and used PlanetLab to run clients accessing the service from different countries. We observed that our protocols improved the perceived availability and, in most cases, the access latency when compared with cloud providers individually. Moreover, the monetary costs of using DEPSKY on this scenario is twice the cost of using a single cloud, which is optimal and seems to be a reasonable cost, given the benefits.

**“Deniable encryption with negligible detection probability: An interactive construction,”**

M. Dürmuth and D. M. Freeman

*Deniable encryption*, introduced in 1997 by Canetti, Dwork, Naor, and Ostrovsky, guarantees that the sender or the receiver of a secret message is able to “fake” the message encrypted in a specific ciphertext in the presence of a coercing adversary, without the adversary detecting that he was not given the real message. To date, constructions are only known either for weakened variants with separate “honest” and “dishonest” encryption algorithms, or for single-algorithm schemes with non-negligible detection probability.

We propose the first sender-deniable public key encryption system with a single encryption algorithm and negligible detection probability. We describe a generic interactive construction based on a public key bit encryption scheme that has certain properties, and we give two examples of encryption schemes with these properties, one based on the quadratic residuosity assumption and the other on trapdoor permutations.

**“AONT-RS: Blending Security and Performance in Dispersed Storage Systems,”**

J. K. Resch and J. S. Plank,

Dispersing files across multiple sites yields a variety of obvious benefits, such as availability, proximity and reliability. Less obviously, it enables security to be achieved without relying on encryption keys. Standard approaches to dispersal either achieve very high security with correspondingly high computational and storage costs, or low security with lower costs. In this paper, we describe a new dispersal scheme, called AONT-RS, which blends an All-Or-Nothing Transform with Reed-Solomon coding to achieve high security with low computational and storage costs. We evaluate this scheme both theoretically and as implemented with standard open source tools. AONTRS forms the backbone of a commercial dispersed storage system, which we briefly describe and then use as a further experimental testbed. We conclude with details of actual deployments.

**3. FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**3.1 ECONOMICAL FEASIBILITY:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**3.2 TECHNICAL FEASIBILITY:**

` This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**3.3 SOCIAL FEASIBILITY:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**4.SYSTEM REQUIREMENTS**

**4.1 INTRODUCTION:**

System analysis is first stage according to system development life cycle model.This system analysis is a process that starts with analyst. Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of a system . One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. During analysis, data are collected for available files , decision points , and transactions handled by the present system.Logical system models and tools that are used in analysis . Training ,experience , and common sense are required for collection of information needed for analysis.

**H/W System Configuration: -**

|  |  |
| --- | --- |
| * Alexa | * Alexa enabled Echo devices or Android or iOS |

**S/W System Configuration: -**

|  |  |
| --- | --- |
| * Operating System | * Android, Linux, Windows, |
| * Front End | * Alexa |
| * Hosting Server | * AWS lambda |
| * Back End | * Python, |
| * Server Framework | * Flask |
| * Database | * Cloud Firestore by Firebase |
| * Language | * Python 3.6 (little bit old <3) |
| * Scripting Language * Tunneling Software | * Json * Ngrok |

**5. SYSTEM DESIGN**

**5.1 INTRODUCTION:**

The most quick witted and obstinate appointment of the animation return is jurisprudence outline. The style chunk describes an exhaustive encrypt and the enterprise by which it is ripe. It refers to the polytechnic specification walk spinal column be realistic in implementation the runner corpus juries. The designers want is none the less the collect is to be concern and in what aim samples of the take in and input are conversion then presented.

**5.2 UML DIAGRAMS:**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

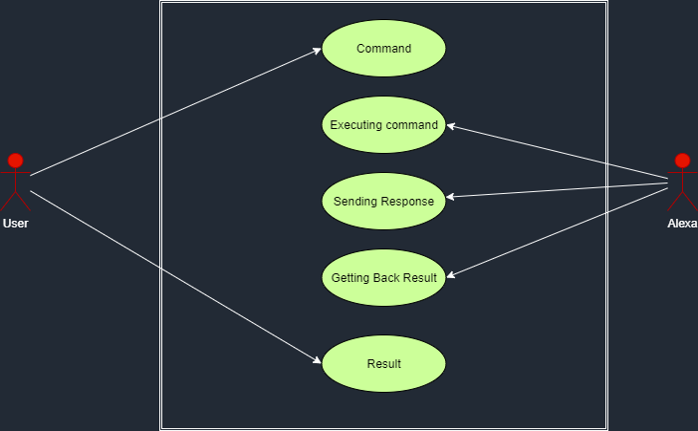
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

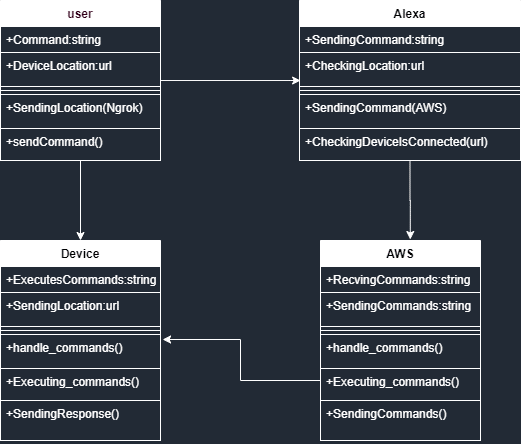
**5.2.1 USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

****

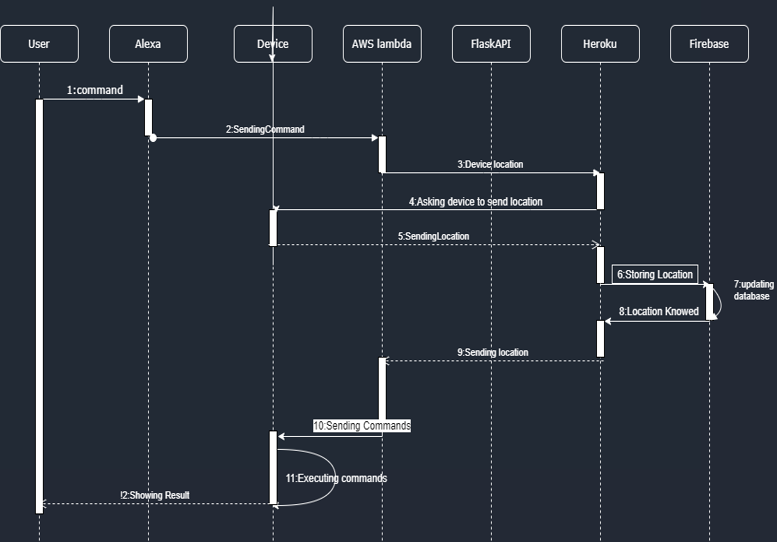
**5.2.2 CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



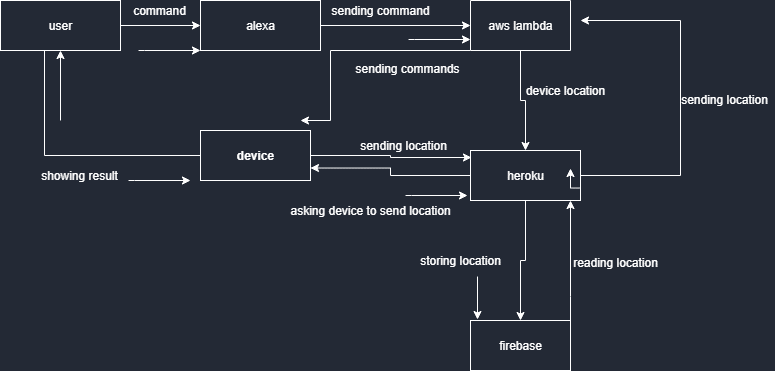
**5.2.3 SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



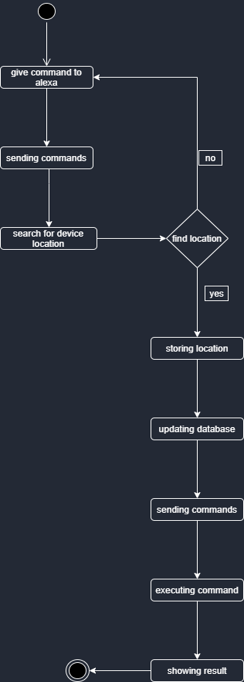
**5.2.4 COLLABORATION DIAGRAM:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the objects



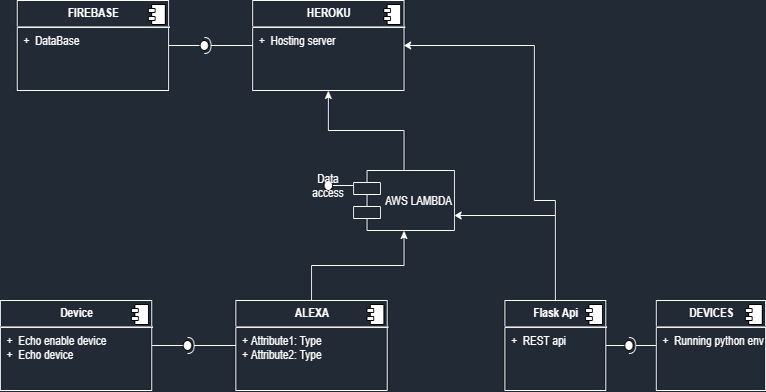
**5.2.5 ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

****

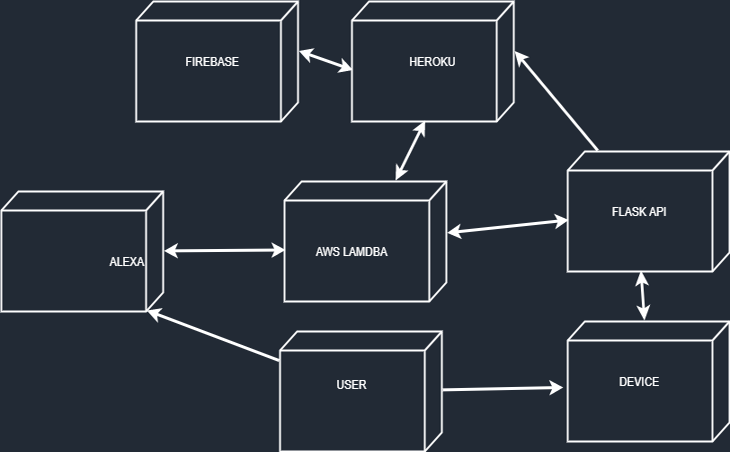
**5.2.6 COMPONENT DIAGRAM:**

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executables, libraries etc. So the purpose of this diagram is different, Component diagrams are used during the implementation phase of an application. But it is prepared well in advance to visualize the implementation details. Initially the system is designed using different UML diagrams and then when the artifacts are ready component diagrams are used to get an idea of the implementation.

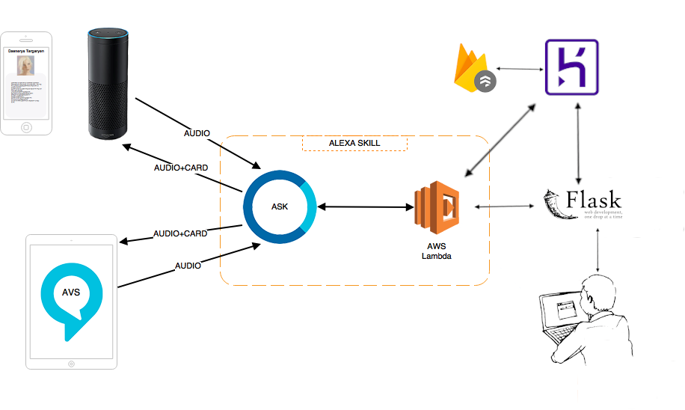


**5.2.7 DEPLOYMENT DIAGRAM:**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.



**5.3 SYSTEM ARCHITECYURE:**



|  |  |
| --- | --- |
|  | Alexa is Amazon’s cloud-based voice assistant that enables the functioning of the Amazon Echo. |
|  | Heroku is a cloud platform as a service supporting several programming languages.. |
|  | AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services. |
|  | Cloud Fire store is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. network latency or Internet connectivity. |
|  | Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. |
|  | Python is an interpreted, high-level, general-purpose programming language. |

**6. IMPLEMENTATION**

**6.1: INTODUCTION:**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve change over and evaluation of change over methods.

**6.2: MODULE DESCRIPTION:**

1. **Alexa**
2. **AWS Lambda**
3. **Heroku**
   1. **Firebase (Fire store)**
4. **4.FlaskApi**
5. **5.Ngrok**

**6.2.1. Alexa:**

In this module first we have to develop an Alexa skill in Alexa Developer console. Then we have to get ARN token form the AWS lambda server. Which is the end point for Alexa and to process Alexa skill information which as taken form the user. Here Alexa act as intermediate between AWS lambda and user

**6.2.2. AWS lambda:**

In this module we have deploy a Python function to AWS lambda. so, it can response to the request made by Alexa. Python Function which is deployed in Lambda is heart of whole process where it handles all the errors, exceptions which the Alexa wants to get back as response massage to show to the user

**6.2.3. Heroku:**

In this module a flask Api is running server which handles all the database function. Main purpose of this is one to know where the is located on internet.

**6.2.3. Firebase (Fire store)**

In this module Heroku take data form the Fire store which has stored in format of json and it can also write to database also

**6.2.4 Flask API**

In this module Flask API which running on local computer which performs all the automation which provided by the admin. This API takes request for NGROK and follow it commands

**6.2.5 NGROK**

In this module Ngrok is tunneling server which connect local server to www server so AWS Lambda can easily place request and take back the response if it provided

**7.CODING**

7.1ALEXA.JSON

1. {
2. "interactionModel": {
3. "languageModel": {
4. "invocationName": "automation skill",
5. "intents": [
6. {
7. "name": "AMAZON.FallbackIntent",
8. "samples": []
9. },
10. {
11. "name": "AMAZON.CancelIntent",
12. "samples": []
13. },
14. {
15. "name": "AMAZON.HelpIntent",
16. "samples": []
17. },
18. {
19. "name": "AMAZON.StopIntent",
20. "samples": []
21. },
22. {
23. "name": "AMAZON.NavigateHomeIntent",
24. "samples": []
25. },
26. {
27. "name": "WhatsMyCommandIntent",
28. "slots": [],
29. "samples": [
30. "last commands"
31. ]
32. },
33. {
34. "name": "ThingsYouCanDo",
35. "slots": [],
36. "samples": [
37. "what can u do for me",
38. "List the commands",
39. "Tell me what u can do",
40. "What you can do"
41. ]
42. },
43. {
44. "name": "ExecutingCommands",
45. "slots": [
46. {
47. "name": "SingleCommands",
48. "type": "singlelinecommands"
49. }
50. ],
51. "samples": [
52. "can u show my {SingleCommands} ",
53. "open {SingleCommands}",
54. "can u {SingleCommands}",
55. "arrange the {SingleCommands}",
56. "run the {SingleCommands}",
57. "clean the {SingleCommands}",
58. "start {SingleCommands}",
59. "can u start {SingleCommands}",
60. "{SingleCommands}"
61. ]
62. },
63. {
64. "name": "OpeingNormalPrograms",
65. "slots": [
66. {
67. "name": "NormalProgram",
68. "type": "ProgramNames"
69. }
70. ],
71. "samples": [
72. "run {NormalProgram}",
73. "start {NormalProgram}",
74. "open {NormalProgram}"
75. ]
76. },
77. {
78. "name": "DialogCommands",
79. "slots": [
80. {
81. "name": "commands",
82. "type": "commands"
83. },
84. {
85. "name": "projectname",
86. "type": "AMAZON.SearchQuery",
87. "samples": [
88. "name of the project is {projectname}",
89. "name is {projectname}",
90. "project name is {projectname}",
91. "{projectname}"
92. ]
93. }
94. ],
95. "samples": [
96. "{commands}",
97. "create {commands}",
98. "can create {commands} "
99. ]
100. },
101. {
102. "name": "GettingResult",
103. "slots": [
104. {
105. "name": "year",
106. "type": "year",
107. "samples": [
108. "year",
109. "{year} of Btech",
110. "{year} year"
111. ]
112. },
113. {
114. "name": "sem",
115. "type": "Sem",
116. "samples": [
117. "{sem} sem",
118. "{sem} of Btech",
119. "{sem}"
120. ]
121. },
122. {
123. "name": "ExamYear",
124. "type": "AMAZON.DATE",
125. "samples": [
126. "{ExamYear}"
127. ]
128. }
129. ],
130. "samples": [
131. "can u show my result",
132. "get my result",
133. "show my result",
134. "result"
135. ]
136. },
137. {
138. "name": "Loginer",
139. "slots": [],
140. "samples": [
141. "open facebook",
142. "login my facebook"
143. ]
144. }
145. ],
146. "types": [
147. {
148. "name": "singlelinecommands",
149. "values": [
150. {
151. "name": {
152. "value": "screenmirror",
153. "synonyms": [
154. "screen mirroring",
155. "screenmirroring",
156. "screen",
157. "screen share"
158. ]
159. }
160. },
161. {
162. "name": {
163. "value": "desktopCleaner",
164. "synonyms": [
165. "desktop",
166. "cleaner",
167. "desktop cleaner"
168. ]
169. }
170. }
171. ]
172. },
173. {
174. "name": "ProgramNames",
175. "values": [
176. {
177. "name": {
178. "value": "cacl",
179. "synonyms": [
180. "calculator"
181. ]
182. }
183. },
184. {
185. "name": {
186. "value": "start chrome",
187. "synonyms": [
188. "chrome",
189. "browser"
190. ]
191. }
192. },
193. {
194. "name": {
195. "value": "notepad"
196. }
197. },
198. {
199. "name": {
200. "value": "code",
201. "synonyms": [
202. "vscode"
203. ]
204. }
205. },
206. {
207. "name": {
208. "value": "mspaint",
209. "synonyms": [
210. "paint"
211. ]
212. }
213. }
214. ]
215. },
216. {
217. "name": "commands",
218. "values": [
219. {
220. "name": {
221. "value": "flutterproject",
222. "synonyms": [
223. "flutter project",
224. "project",
225. "flutter"
226. ]
227. }
228. }
229. ]
230. },
231. {
232. "name": "Sem",
233. "values": [
234. {
235. "name": {
236. "value": "II",
237. "synonyms": [
238. "second",
239. "two",
240. "2"
241. ]
242. }
243. },
244. {
245. "name": {
246. "value": "I",
247. "synonyms": [
248. "frist",
249. "one",
250. "1"
251. ]
252. }
253. }
254. ]
255. },
256. {
257. "name": "year",
258. "values": [
259. {
260. "name": {
261. "value": "IV",
262. "synonyms": [
263. "fourth",
264. "4"
265. ]
266. }
267. },
268. {
269. "name": {
270. "value": "III",
271. "synonyms": [
272. "third",
273. "3"
274. ]
275. }
276. },
277. {
278. "name": {
279. "value": "II",
280. "synonyms": [
281. "two",
282. "2"
283. ]
284. }
285. },
286. {
287. "name": {
288. "value": "I",
289. "synonyms": [
290. "one",
291. "1"
292. ]
293. }
294. }
295. ]
296. },
297. {
298. "name": "namesofapps",
299. "values": [
300. {
301. "name": {
302. "value": "fb"
303. }
304. },
305. {
306. "name": {
307. "value": "facebook"
308. }
309. }
310. ]
311. }
312. ]
313. },
314. "dialog": {
315. "intents": [
316. {
317. "name": "ExecutingCommands",
318. "delegationStrategy": "ALWAYS",
319. "confirmationRequired": true,
320. "prompts": {
321. "confirmation": "Confirm.Intent.398462502021"
322. },
323. "slots": [
324. {
325. "name": "SingleCommands",
326. "type": "singlelinecommands",
327. "confirmationRequired": false,
328. "elicitationRequired": false,
329. "prompts": {}
330. }
331. ]
332. },
333. {
334. "name": "DialogCommands",
335. "confirmationRequired": true,
336. "prompts": {
337. "confirmation": "Confirm.Intent.1303764452968"
338. },
339. "slots": [
340. {
341. "name": "commands",
342. "type": "commands",
343. "confirmationRequired": false,
344. "elicitationRequired": false,
345. "prompts": {}
346. },
347. {
348. "name": "projectname",
349. "type": "AMAZON.SearchQuery",
350. "confirmationRequired": true,
351. "elicitationRequired": true,
352. "prompts": {
353. "confirmation": "Confirm.Slot.1303764452968.1214890330054",
354. "elicitation": "Elicit.Slot.763323224860.769562388878"
355. }
356. }
357. ]
358. },
359. {
360. "name": "GettingResult",
361. "confirmationRequired": false,
362. "prompts": {},
363. "slots": [
364. {
365. "name": "year",
366. "type": "year",
367. "confirmationRequired": true,
368. "elicitationRequired": true,
369. "prompts": {
370. "confirmation": "Confirm.Slot.362732585282.1351795572908",
371. "elicitation": "Elicit.Slot.362732585282.1351795572908"
372. }
373. },
374. {
375. "name": "sem",
376. "type": "Sem",
377. "confirmationRequired": true,
378. "elicitationRequired": true,
379. "prompts": {
380. "confirmation": "Confirm.Slot.362732585282.109410562451",
381. "elicitation": "Elicit.Slot.362732585282.109410562451"
382. }
383. },
384. {
385. "name": "ExamYear",
386. "type": "AMAZON.DATE",
387. "confirmationRequired": true,
388. "elicitationRequired": true,
389. "prompts": {
390. "confirmation": "Confirm.Slot.362732585282.1446529009495",
391. "elicitation": "Elicit.Slot.362732585282.1446529009495"
392. }
393. }
394. ]
395. }
396. ],
397. "delegationStrategy": "ALWAYS"
398. },
399. "prompts": [
400. {
401. "id": "Confirm.Intent.898526131785",
402. "variations": [
403. {
404. "type": "PlainText",
405. "value": "can tell project name"
406. }
407. ]
408. },
409. {
410. "id": "Elicit.Slot.763323224860.769562388878",
411. "variations": [
412. {
413. "type": "PlainText",
414. "value": "can tell the name of project"
415. },
416. {
417. "type": "PlainText",
418. "value": "project name ?"
419. },
420. {
421. "type": "PlainText",
422. "value": "can you say project name"
423. },
424. {
425. "type": "PlainText",
426. "value": "what is project name"
427. }
428. ]
429. },
430. {
431. "id": "Elicit.Slot.763323224860.1442318176051",
432. "variations": [
433. {
434. "type": "PlainText",
435. "value": "project name please?"
436. },
437. {
438. "type": "PlainText",
439. "value": "can you tell project name"
440. }
441. ]
442. },
443. {
444. "id": "Elicit.Slot.362732585282.1351795572908",
445. "variations": [
446. {
447. "type": "PlainText",
448. "value": "Which year of Btech do you want?"
449. }
450. ]
451. },
452. {
453. "id": "Confirm.Slot.362732585282.1351795572908",
454. "variations": [
455. {
456. "type": "PlainText",
457. "value": "Is it {year} year Btech,right?"
458. }
459. ]
460. },
461. {
462. "id": "Elicit.Slot.362732585282.109410562451",
463. "variations": [
464. {
465. "type": "PlainText",
466. "value": "Which Sem of {year} year Btech"
467. }
468. ]
469. },
470. {
471. "id": "Confirm.Slot.362732585282.109410562451",
472. "variations": [
473. {
474. "type": "PlainText",
475. "value": "is it {sem} sem of {year} year btech , right?"
476. }
477. ]
478. },
479. {
480. "id": "Confirm.Slot.1303764452968.1214890330054",
481. "variations": [
482. {
483. "type": "PlainText",
484. "value": "{projectname} .is it correct?"
485. }
486. ]
487. },
488. {
489. "id": "Confirm.Intent.1303764452968",
490. "variations": [
491. {
492. "type": "PlainText",
493. "value": "Do u want to create a flutter project with name {projectname} , is it right?"
494. }
495. ]
496. },
497. {
498. "id": "Elicit.Slot.362732585282.1446529009495",
499. "variations": [
500. {
501. "type": "PlainText",
502. "value": "in which year do u want "
503. }
504. ]
505. },
506. {
507. "id": "Confirm.Slot.362732585282.1446529009495",
508. "variations": [
509. {
510. "type": "PlainText",
511. "value": "Btech {year} year of {sem} sem in {ExamYear} . is it right "
512. }
513. ]
514. },
515. {
516. "id": "Confirm.Intent.398462502021",
517. "variations": [
518. {
519. "type": "PlainText",
520. "value": "is it {SingleCommands} do want to run?"
521. }
522. ]
523. }
524. ]
525. }
526. }

7.2lambda\_function.py

1. """
2. This sample demonstrates a simple skill built with the Amazon Alexa Skills Kit.
3. The Intent Schema, Custom Slots, and Sample Utterances for this skill, as well
4. as testing instructions are located at http://amzn.to/1LzFrj6
5. For additional samples, visit the Alexa Skills Kit Getting Started guide at
6. http://amzn.to/1LGWsLG
7. """
9. from \_\_future\_\_ import print\_function
10. import requests
11. import calendar

14. def lambda\_handler(event, context):
15. """ Route the incoming request based on type (LaunchRequest, IntentRequest,
16. etc.) The JSON body of the request is provided in the event parameter.
17. """
18. print("event.session.application.applicationId=" +
19. event['session']['application']['applicationId'])
20. print(" printing context")
21. print(context)
23. """
24. Uncomment this if statement and populate with your skill's application ID to
25. prevent someone else from configuring a skill that sends requests to this
26. function.
27. """
28. # if (event['session']['application']['applicationId'] !=
29. # "amzn1.echo-sdk-ams.app.[unique-value-here]"):
30. # raise ValueError("Invalid Application ID")
32. if event['session']['new']:
33. on\_session\_started({'requestId': event['request']['requestId']},
34. event['session'])
36. if event['request']['type'] == "LaunchRequest":
37. return on\_launch(event['request'], event['session'])
38. elif event['request']['type'] == "IntentRequest":
39. return on\_intent(event['request'], event['session'])
40. elif event['request']['type'] == "SessionEndedRequest":
41. return on\_session\_ended(event['request'], event['session'])

44. def on\_session\_started(session\_started\_request, session):
45. """ Called when the session starts """
47. print("on\_session\_started requestId=" + session\_started\_request['requestId']
48. + ", sessionId=" + session['sessionId'])

51. def on\_launch(launch\_request, session):
52. """ Called when the user launches the skill without specifying what they
53. want
54. """
56. print("on\_launch requestId=" + launch\_request['requestId'] +
57. ", sessionId=" + session['sessionId'])
58. # Dispatch to your skill's launch
59. return get\_welcome\_response()

62. def on\_intent(intent\_request, session):
63. """ Called when the user specifies an intent for this skill """
65. print("on\_intent requestId=" + intent\_request['requestId'] +
66. ", sessionId=" + session['sessionId'])
68. intent = intent\_request['intent']
69. intent\_name = intent\_request['intent']['name']
70. print(intent\_name)
71. # Dispatch to your skill's intent handlers
72. if intent\_name == "ThingsYouCanDo":
73. return list\_commands()
74. elif intent\_name =="ExecutingCommands":
75. return sending\_command\_to\_device(intent,session)
76. elif intent\_name == "DialogCommands":
77. return dialogCommand(intent,session)
78. elif intent\_name == "AMAZON.HelpIntent":
79. return get\_welcome\_response()
80. elif intent\_name=="WhatsMyCommandIntent":
81. return get\_command\_from\_session(intent,session)
82. elif intent\_name=="OpeingNormalPrograms":
83. return sending\_command\_to\_device(intent,session)
84. elif intent\_name=="AMAZON.FallbackIntent":
85. return help(intent,session)
86. elif intent\_name=="GettingResult":
87. return result(intent,session)
88. elif intent\_name=="Loginer":
89. return loginer(intent,session)
90. elif intent\_name=="AMAZON.StopIntent":
91. return on\_session\_ended(intent, session)
92. else:
93. print(intent\_name)
94. raise ValueError("Invalid intent")

97. def on\_session\_ended(session\_ended\_request, session):
98. """ Called when the user ends the session.
99. Is not called when the skill returns should\_end\_session=true
100. """
101. # print("on\_session\_ended requestId=" + session\_ended\_request['requestId'] +
102. # ", sessionId=" + session['sessionId'])
103. session\_attributes = {}
104. card\_title = "Error"
105. speech\_output = "Thanking for using our skill"
107. # If the user either does not reply to the welcome message or says something
108. # that is not understood, they will be prompted again with this text.
109. reprompt\_text = "come back gain"
110. should\_end\_session = False
111. return build\_response(session\_attributes, build\_speechlet\_response(
112. card\_title, speech\_output, reprompt\_text, should\_end\_session))
113. # add cleanup logic here
115. # --------------- Functions that control the skill's behavior ------------------

118. def get\_welcome\_response():
119. """ If we wanted to initialize the session to have some attributes we could
120. add those here
121. """
122. url=getting\_device\_location()
123. session\_attributes = {}
124. card\_title = "Welcome"
125. html = requests.get(url)
126. speech\_output = "Welcome to automation feel good to intract with us. "+html.text
128. # If the user either does not reply to the welcome message or says something
129. # that is not understood, they will be prompted again with this text.
130. reprompt\_text = "say , create project or Get results or clean my Desktop"
131. should\_end\_session = False
132. return build\_response(session\_attributes, build\_speechlet\_response(
133. card\_title, speech\_output, reprompt\_text, should\_end\_session))

136. def set\_command\_in\_session(intent, session):
137. """ Sets the command in the session and prepares the speech to reply to the
138. user.
139. """
141. card\_title = intent['name']
142. session\_attributes = {}
143. should\_end\_session = False
145. if 'Command' in intent['slots']:
146. favorite\_command = intent['slots']['Command']['value']
147. session\_attributes = create\_favorite\_command\_attributes(favorite\_command)
148. html = requests.get('http://example.ngrok.io/command?command='+favorite\_command)
149. speech\_output = "I sent the command to your system.Let me know if you want me to send another command."
150. reprompt\_text = "Please tell me the command I should send to your system by saying, " \
151. "Send the shutdown command"
152. else:
153. speech\_output = "I did not understand that. Please try again."
154. reprompt\_text = "Please tell me the command I should send to your system by saying, " \
155. "Send the shutdown command"

158. return build\_response(session\_attributes, build\_speechlet\_response(
159. card\_title, speech\_output, reprompt\_text, should\_end\_session))


163. def create\_favorite\_command\_attributes(favorite\_command):
164. return {"favoriteCommand": favorite\_command}

167. def help(intent,session):
168. session\_attributes = {}
169. card\_title = "Error"
170. speech\_output = "unable to understand you request"
172. # If the user either does not reply to the welcome message or says something
173. # that is not understood, they will be prompted again with this text.
174. reprompt\_text = "say , create flutter project or Get results or clean my Desktop \n To know more command say , what can you do for me"
175. should\_end\_session = False
176. return build\_response(session\_attributes, build\_speechlet\_response(
177. card\_title, speech\_output, reprompt\_text, should\_end\_session))

180. def dialogCommand(intent,session):
182. url=getting\_device\_location()
184. passingcommand=gettingvalueFormintent(intent['slots']['commands'])
185. arguments=intent['slots']['projectname']['value']
186. favorite\_command =intent['slots']['commands']['value']
187. card\_title =intent['name']
188. # print(url+"/dialogcommand?command="+passingcommand+"&args="+arguments)
189. html = requests.get(url+"/dialogcommand?command="+passingcommand+"&args="+arguments)
190. should\_end\_session=False
191. speech\_output='Executing '+html.text
192. reprompt\_text="i think it is Executed"
194. session\_attributes=create\_favorite\_command\_attributes(favorite\_command)
195. return build\_response(session\_attributes, build\_speechlet\_response(
196. card\_title, speech\_output, reprompt\_text, should\_end\_session))

199. def get\_command\_from\_session(intent, session):
200. session\_attributes = {}
201. reprompt\_text = None
203. if "favoriteCommand" in session.get('attributes', {}):
204. favorite\_command = session['attributes']['favoriteCommand']
205. speech\_output = "Your last command was " + favorite\_command + \
206. ". Goodbye."
207. should\_end\_session = True
208. else:
209. speech\_output = "I'm not sure what your last command was. " \
210. "Please tell me the command I should send to your system by saying, " \
211. "Send the shutdown command"
212. should\_end\_session = False
214. # Setting reprompt\_text to None signifies that we do not want to reprompt
215. # the user. If the user does not respond or says something that is not
216. # understood, the session will end.
217. return build\_response(session\_attributes, build\_speechlet\_response(
218. intent['name'], speech\_output, reprompt\_text, should\_end\_session))

221. def list\_commands():
222. session\_attributes = {}
223. url=getting\_device\_location()
224. card\_title = "Commands"
225. html = requests.get(url+"/command?command=getcommands")
226. speech\_output = f"{html.text}"
227. # If the user either does not reply to the welcome message or says something
228. # that is not understood, they will be prompted again with this text.
229. reprompt\_text = "say , Create project \nDesktop Cleaner \nget Results"
230. should\_end\_session = False
231. return build\_response(session\_attributes, build\_speechlet\_response(
232. card\_title, speech\_output, reprompt\_text, should\_end\_session))

235. def sending\_command\_to\_device(intent, session):
236. url=getting\_device\_location()
237. session\_attributes = {}
239. if 'NormalProgram' in intent['slots'] :
240. favorite\_command =intent['slots']['NormalProgram']['value']
241. card\_title =intent['name']
242. value=gettingvalueFormintent(intent['slots']['NormalProgram'])
243. should\_end\_session=False
244. session\_attributes=create\_favorite\_command\_attributes(favorite\_command)
245. url = getting\_device\_location\_form\_tmp()
246. html = requests.get(url+"/command?command="+value)
247. speech\_output=html.text
248. reprompt\_text="i think it is openend please check it"
250. elif 'SingleCommands' in intent['slots']:
251. favorite\_command =intent['slots']['SingleCommands']['value']
252. value=gettingvalueFormintent(intent['slots']['SingleCommands'])
253. card\_title =intent['name']
254. should\_end\_session=False
255. session\_attributes=create\_favorite\_command\_attributes(favorite\_command)
256. print(url)
257. html = requests.get(url+"/command?command="+value)
258. speech\_output='Executing ' +favorite\_command
259. reprompt\_text="i think it is Executed"
260. else :
261. card\_title="error"
262. speech\_output="unable to process ur request"
263. should\_end\_session=True
264. return build\_response(session\_attributes, build\_speechlet\_response(
265. card\_title, speech\_output, reprompt\_text, should\_end\_session))

268. def result(intent, session):
269. url=getting\_device\_location()
270. passingcommand="result"
271. yearwithmonth=""
272. Btechyear=""
273. Btechsem=""
274. finalvalue=""
275. examyear=intent['slots']['ExamYear']['value']
276. Btechsem=gettingvalueFormintent(intent['slots']['sem'])
277. Btechyear=gettingvalueFormintent(intent['slots']['year'])
278. eyear=examyear.split("-")[0]
279. if eyear=="2019":
280. monthabbr=calendar.month\_abbr[int(examyear.split("-")[1])]
281. if monthabbr=="Nov":
282. yearwithmonth=f'"Nov/Dec {eyear}"'
283. elif monthabbr=="June" or monthabbr=="Jun":
284. yearwithmonth=f'"June/July {eyear}"'
285. elif monthabbr=="July"or monthabbr=="Jul":
286. yearwithmonth=f'"June/July {eyear}"'
287. elif monthabbr=="Dec":
288. yearwithmonth=f'"Nov/Dec {eyear}"'
289. elif monthabbr==calendar.month\_abbr[4]:
290. yearwithmonth=f'"April {eyear}"'
291. elif eyear=="2018":
292. monthabbr=calendar.month\_name[int(examyear.split("-")[1])]
293. yearwithmonth=f'"{monthabbr} {eyear}"'
294. finalvalue=f'"B.Tech {Btechyear} Year {Btechsem} Semester (R15)"'
295. print(finalvalue)
296. print(yearwithmonth)
297. favorite\_command ="result"
298. card\_title =intent['name']
299. # print(url+"/dialogcommand?command="+passingcommand+"&args="+arguments)
300. html = requests.get(url+"/result?value1="+finalvalue+"&value2="+yearwithmonth)
301. should\_end\_session=False
302. speech\_output='Executing '+html.text
303. # speech\_output= finalvalue +" "+yearwithmonth
304. reprompt\_text="i think it is Executed"
306. session\_attributes=create\_favorite\_command\_attributes(favorite\_command)
307. return build\_response(session\_attributes, build\_speechlet\_response(
308. card\_title, speech\_output, reprompt\_text, should\_end\_session))

311. def loginer(intent, session):
312. url=getting\_device\_location()
313. favorite\_command ="Loginer"
314. card\_title =intent['name']
315. # print(url+"/dialogcommand?command="+passingcommand+"&args="+arguments)
316. html = requests.get(url+"/loginer")
317. should\_end\_session=False
318. speech\_output=''+html.text
319. # speech\_output= finalvalue +" "+yearwithmonth
320. reprompt\_text="i think it is Executed"
322. session\_attributes=create\_favorite\_command\_attributes(favorite\_command)
323. return build\_response(session\_attributes, build\_speechlet\_response(
324. card\_title, speech\_output, reprompt\_text, should\_end\_session))

327. # --------------- Function for getting location of devcies ----------------------#
328. def getting\_device\_location():
329. url=requests.get("https://alexautomation.herokuapp.com/read?id=url")
330. with open('/tmp/url.txt', 'w') as file:
331. file.write(url.text)
332. return url.text

335. def getting\_device\_location\_form\_tmp():
336. try:
337. with open('/tmp/url.txt','r') as file:
338. gobal=file.readline()
339. if gobal!="":
340. # print(gobal)
341. return gobal
342. else :
343. getting\_Device\_location()
344. except:
345. print("identifier")
346. # --------------- Helpers that build all of the responses ----------------------#

349. def build\_speechlet\_response(title, output, reprompt\_text, should\_end\_session):
350. return {
351. 'outputSpeech': {
352. 'type': 'PlainText',
353. 'text': output
354. },
355. 'card': {
356. 'type': 'Simple',
357. 'title': 'WeRnoobs - ' + title,
358. 'content': output
359. },
360. 'reprompt': {
361. 'outputSpeech': {
362. 'type': 'PlainText',
363. 'text': reprompt\_text
364. }
365. },
366. 'shouldEndSession': should\_end\_session
367. }

370. def build\_response(session\_attributes, speechlet\_response):
371. return {
372. 'version': '1.0',
373. 'sessionAttributes': session\_attributes,
374. 'response': speechlet\_response
375. }
376. #--- utils--#
377. def gettingvalueFormintent(intent):
378. value=""
379. try:
380. for i in intent['resolutions']['resolutionsPerAuthority']:
381. a=i['values']
382. for i in a:
383. value=i['value']['name']
384. except Exception as e:
385. return "open dropbox"
387. return value

7.3.app.py

1. from flask import Flask, escape, request, jsonify,render\_template
2. from firebase\_admin import credentials, firestore, initialize\_app
3. import json
4. cred = credentials.Certificate("key.json")
5. default\_app = initialize\_app(cred)
6. db = firestore.client()
7. todo\_ref = db.collection('urls')
9. app = Flask(\_\_name\_\_)

12. @app.route('/')
13. def index():
14. return render\_template("index.html")

17. @app.route('/command', methods=['GET'])
18. def handle\_command():
19. command = request.args.get('command', '')
20. return command

23. @app.route('/add', methods=['POST'])
24. def create():
25. # http://300bf87b.ngrok.io/add?url=https://google.in
26. try:
27. \_urls = str(request.args.get('url', ''))
28. \_data = {u"url": \_urls}
29. todo\_ref.document("url").set(\_data)
30. t = todo\_ref.document("url").get()
31. print("checking")
32. # read()
33. if t.to\_dict() == \_data:
34. return jsonify({"success": True}), 200
35. else:
36. return jsonify({"data not entered correctly": False}), 400
37. except Exception as e:
38. return f"An Error Occured: {e}"

41. @app.route('/read', methods=['GET'])
42. def read():
43. # http://300bf87b.ngrok.io/read?id=url
45. try:
46. todo\_id = request.args.get('id')
47. print(todo\_id)
48. if todo\_id:
49. todo = todo\_ref.document(todo\_id).get()
50. # y=todo.to\_dict()
51. print(todo.to\_dict()["url"])
52. return todo.to\_dict()["url"], 200
53. else:
54. all\_todos = [doc.to\_dict() for doc in todo\_ref.stream()]
55. return jsonify(all\_todos), 200
56. except Exception as e:
57. return f"An Error Occured: {e}"

60. @app.route('/delete', methods=['GET', 'DELETE'])
61. def delete():
62. # http://300bf87b.ngrok.io/delete?id=url
63. try:
64. todo\_id = request.args.get('id')
65. todo\_ref.document(todo\_id).delete()
66. return jsonify({"success": True}), 200
67. except Exception as e:
68. return f"An Error Occured: {e}"

71. if \_\_name\_\_ == "\_\_main\_\_":
72. app.jinja\_env.auto\_reload = True
73. app.config['TEMPLATES\_AUTO\_RELOAD'] = True
74. app.run(debug=True)

7.4.CLIENT API

1. from flask import Flask, escape, request
2. import os
3. app = Flask(\_\_name\_\_)
4. import getpass
5. from Executing import ExecutingCommands as EC

8. @app.route('/')
9. def hello():
11. # name=getpass.getuser()
12. name = 'prakash'
13. return f'Hello, {escape(name)}!'

16. @app.route('/command', methods=['GET'])
17. def handle\_command():
18. command = request.args.get('command','')
19. print(command)
20. return EC.handle\_Commands(command)

23. @app.route('/dialogcommand', methods=['GET'])
24. def handle\_dialogcommand():
25. command = request.args.get('command','').replace(" ","")
26. args=request.args.get('args','').replace(" ","")
27. print(command)
28. return EC.DialogCommand(command,args)
30. @app.route('/result', methods=['GET'])
31. def handle\_result():
32. args1 = request.args.get('value1', '')
33. args2 = request.args.get('value2', '')
34. print(args1+args2)
35. return EC.result(args1, args2)
37. @app.route('/loginer', methods=['GET'])
38. def handle\_loginer():
39. EC.loginer()
40. return "Logined"
42. if \_\_name\_\_ == "\_\_main\_\_":
43. app.jinja\_env.auto\_reload = True
44. app.config['TEMPLATES\_AUTO\_RELOAD'] = True
45. app.run(debug=True)

7.5Executing\_command.py

1. import os
2. import os.path
3. import shutil
4. def handle\_Commands(command):
5. #-----opeing file to check the where it suppoerts are not-----#
6. with open("Executing//WindowsApps.txt","r") as \_file:
7. \_WindowsApps=\_file.read()
8. with open("Executing//automatoins.txt") as \_file:
9. \_automation=\_file.read()
11. #-----Main function-----#
12. if command in \_WindowsApps:
13. os.popen(command)
14. return "opened"
15. elif command in \_automation:
16. print(command)
17. return automation(command.replace(" ",""))
18. elif command == "getcommands":
19. return listingCommands(command.replace(" ",""))
20. else:
21. return "i can't able to do that for now. i'm still in beta"
22. return "opened"

25. def automation(command):
26. pwd=os.getcwd()
27. os.popen(pwd+"\\Executing\\"+command+".lnk")
28. return f"{command} started"
30. def DialogCommand(command,args):
31. pwd=os.getcwd()
32. os.popen(pwd+"\\Executing\\"+command+".lnk "+ args.lower())
33. return f"{args} {command} started"
35. def listingCommands(command):
36. with open("Executing/AlexaCando.txt") as \_file:
37. data =\_file.read()
38. return str(data)
40. def result(value1,value2):
41. pwd = os.getcwd()
42. final="python "+pwd+"\\Executing\\Rantest1.py"+" "+value1+" "+value2
43. print(final)
44. os.system(final)
45. return str(value1+value2)
47. def loginer():
48. pwd = os.getcwd()
49. final="python "+pwd+"\\Executing\\Loginer.py"
50. os.system(final)
51. return "logined"

7.6DesktopCleaner.py

1. import os
2. import shutil
4. UserProfile = os.environ.get('USERPROFILE')
6. folder\_to\_track = f'{UserProfile}\\Desktop\\'
8. os.chdir(folder\_to\_track)

11. extensions\_folders = {
12. # No name
13. 'noname': f"{UserProfile}\\Desktop\\Other\\Uncategorized",
14. # Audio
15. '.aif': f"{UserProfile}\\Desktop\\Media\\Audio",
16. '.cda': f"{UserProfile}\\Desktop\\Media\\Audio",
17. '.mid': f"{UserProfile}\\Desktop\\Media\\Audio",
18. '.midi': f"{UserProfile}\\Desktop\\Media\\Audio",
19. '.mp3': f"{UserProfile}\\Desktop\\Media\\Audio",
20. '.mpa': f"{UserProfile}\\Desktop\\Media\\Audio",
21. '.ogg': f"{UserProfile}\\Desktop\\Media\\Audio",
22. '.wav': f"{UserProfile}\\Desktop\\Media\\Audio",
23. '.wma': f"{UserProfile}\\Desktop\\Media\\Audio",
24. '.wpl': f"{UserProfile}\\Desktop\\Media\\Audio",
25. '.m3u': f"{UserProfile}\\Desktop\\Media\\Audio",
26. # Docs
27. '.txt': f"{UserProfile}\\Desktop\\Docs\\TextFiles",
28. '.doc': f"{UserProfile}\\Desktop\\Docs\\Microsoft\\Word",
29. '.docx': f"{UserProfile}\\Desktop\\Docs\\Microsoft\\Word",
30. '.odt ': f"{UserProfile}\\Desktop\\Docs\\TextFiles",
31. '.pdf': f"{UserProfile}\\Desktop\\Docs\\PDF",
32. '.rtf': f"{UserProfile}\\Desktop\\Docs\\TextFiles",
33. '.tex': f"{UserProfile}\\Desktop\\Docs\\TextFiles",
34. '.wks ': f"{UserProfile}\\Desktop\\Docs\\TextFiles",
35. '.wps': f"{UserProfile}\\Desktop\\Docs\\TextFiles",
36. '.wpd': f"{UserProfile}\\Desktop\\Docs\\TextFiles",
37. # Video
38. '.3g2': f"{UserProfile}\\Desktop\\Media\\Video",
39. '.3gp': f"{UserProfile}\\Desktop\\Media\\Video",
40. '.avi': f"{UserProfile}\\Desktop\\Media\\Video",
41. '.flv': f"{UserProfile}\\Desktop\\Media\\Video",
42. '.h264': f"{UserProfile}\\Desktop\\Media\\Video",
43. '.m4v': f"{UserProfile}\\Desktop\\Media\\Video",
44. '.mkv': f"{UserProfile}\\Desktop\\Media\\Video",
45. '.mov': f"{UserProfile}\\Desktop\\Media\\Video",
46. '.mp4': f"{UserProfile}\\Desktop\\Media\\Video",
47. '.mpg': f"{UserProfile}\\Desktop\\Media\\Video",
48. '.mpeg': f"{UserProfile}\\Desktop\\Media\\Video",
49. '.rm': f"{UserProfile}\\Desktop\\Media\\Video",
50. '.swf': f"{UserProfile}\\Desktop\\Media\\Video",
51. '.vob': f"{UserProfile}\\Desktop\\Media\\Video",
52. '.wmv': f"{UserProfile}\\Desktop\\Media\\Video",
53. # Images
54. '.ai': f"{UserProfile}\\Desktop\\Media\\Images",
55. '.bmp': f"{UserProfile}\\Desktop\\Media\\Images",
56. '.gif': f"{UserProfile}\\Desktop\\Media\\Images",
57. '.ico': f"{UserProfile}\\Desktop\\Media\\Images",
58. '.jpg': f"{UserProfile}\\Desktop\\Media\\Images",
59. '.jpeg': f"{UserProfile}\\Desktop\\Media\\Images",
60. '.png': f"{UserProfile}\\Desktop\\Media\\Images",
61. '.ps': f"{UserProfile}\\Desktop\\Media\\Images",
62. '.psd': f"{UserProfile}\\Desktop\\Media\\Images",
63. '.svg': f"{UserProfile}\\Desktop\\Media\\Images",
64. '.tif': f"{UserProfile}\\Desktop\\Media\\Images",
65. '.tiff': f"{UserProfile}\\Desktop\\Media\\Images",
66. '.CR2': f"{UserProfile}\\Desktop\\Media\\Images",
67. # Internet
68. '.asp': f"{UserProfile}\\Desktop\\Other\\Internet",
69. '.aspx': f"{UserProfile}\\Desktop\\Other\\Internet",
70. '.cer': f"{UserProfile}\\Desktop\\Other\\Internet",
71. '.cfm': f"{UserProfile}\\Desktop\\Other\\Internet",
72. '.cgi': f"{UserProfile}\\Desktop\\Other\\Internet",
73. '.pl': f"{UserProfile}\\Desktop\\Other\\Internet",
74. '.css': f"{UserProfile}\\Desktop\\Other\\Internet",
75. '.htm': f"{UserProfile}\\Desktop\\Other\\Internet",
76. '.js': f"{UserProfile}\\Desktop\\Other\\Internet",
77. '.jsp': f"{UserProfile}\\Desktop\\Other\\Internet",
78. '.part': f"{UserProfile}\\Desktop\\Other\\Internet",
79. '.php': f"{UserProfile}\\Desktop\\Other\\Internet",
80. '.rss': f"{UserProfile}\\Desktop\\Other\\Internet",
81. '.xhtml': f"{UserProfile}\\Desktop\\Other\\Internet",
82. # Compressed
83. '.7z': f"{UserProfile}\\Desktop\\Other\\Compressed",
84. '.arj': f"{UserProfile}\\Desktop\\Other\\Compressed",
85. '.deb': f"{UserProfile}\\Desktop\\Other\\Compressed",
86. '.pkg': f"{UserProfile}\\Desktop\\Other\\Compressed",
87. '.rar': f"{UserProfile}\\Desktop\\Other\\Compressed",
88. '.rpm': f"{UserProfile}\\Desktop\\Other\\Compressed",
89. '.tar.gz': f"{UserProfile}\\Desktop\\Other\\Compressed",
90. '.z': f"{UserProfile}\\Desktop\\Other\\Compressed",
91. '.zip': f"{UserProfile}\\Desktop\\Other\\Compressed",
92. # Disc
93. '.bin': f"{UserProfile}\\Desktop\\Other\\Disc",
94. '.dmg': f"{UserProfile}\\Desktop\\Other\\Disc",
95. '.iso': f"{UserProfile}\\Desktop\\Other\\Disc",
96. '.toast': f"{UserProfile}\\Desktop\\Other\\Disc",
97. '.vcd': f"{UserProfile}\\Desktop\\Other\\Disc",
98. # Data
99. '.csv': f"{UserProfile}\\Desktop\\Programming\\Database",
100. '.dat': f"{UserProfile}\\Desktop\\Programming\\Database",
101. '.db': f"{UserProfile}\\Desktop\\Programming\\Database",
102. '.dbf': f"{UserProfile}\\Desktop\\Programming\\Database",
103. '.log': f"{UserProfile}\\Desktop\\Programming\\Database",
104. '.mdb': f"{UserProfile}\\Desktop\\Programming\\Database",
105. '.sav': f"{UserProfile}\\Desktop\\Programming\\Database",
106. '.sql': f"{UserProfile}\\Desktop\\Programming\\Database",
107. '.tar': f"{UserProfile}\\Desktop\\Programming\\Database",
108. '.xml': f"{UserProfile}\\Desktop\\Programming\\Database",
109. '.json': f"{UserProfile}\\Desktop\\Programming\\Database",
110. # Executables
111. '.apk': f"{UserProfile}\\Desktop\\Other\\Executables",
112. '.bat': f"{UserProfile}\\Desktop\\Other\\Executables",
113. '.com': f"{UserProfile}\\Desktop\\Other\\Executables",
114. '.exe': f"{UserProfile}\\Desktop\\Other\\Executables",
115. '.gadget': f"{UserProfile}\\Desktop\\Other\\Executables",
116. '.jar': f"{UserProfile}\\Desktop\\Other\\Executables",
117. '.wsf': f"{UserProfile}\\Desktop\\Other\\Executables",
118. # Fonts
119. '.fnt': f"{UserProfile}\\Desktop\\Other\\Fonts",
120. '.fon': f"{UserProfile}\\Desktop\\Other\\Fonts",
121. '.otf': f"{UserProfile}\\Desktop\\Other\\Fonts",
122. '.ttf': f"{UserProfile}\\Desktop\\Other\\Fonts",
123. # Presentations
124. '.key': f"{UserProfile}\\Desktop\\Docs\\Presentations",
125. '.odp': f"{UserProfile}\\Desktop\\Docs\\Presentations",
126. '.pps': f"{UserProfile}\\Desktop\\Docs\\Presentations",
127. '.ppt': f"{UserProfile}\\Desktop\\Docs\\Presentations",
128. '.pptx': f"{UserProfile}\\Desktop\\Docs\\Presentations",
129. # Programming
130. '.c': f"{UserProfile}\\Desktop\\Programming\\C&C++",
131. '.class': f"{UserProfile}\\Desktop\\Programming\\Java",
132. '.dart': f"{UserProfile}\\Desktop\\Programming\\Dart",
133. '.py': f"{UserProfile}\\Desktop\\Programming\\Python",
134. '.sh': f"{UserProfile}\\Desktop\\Programming\\Shell",
135. '.swift': f"{UserProfile}\\Desktop\\Programming\\Swift",
136. '.html': f"{UserProfile}\\Desktop\\Programming\\C&C++",
137. '.h': f"{UserProfile}\\Desktop\\Programming\\C&C++",
138. # Spreadsheets
139. '.ods': f"{UserProfile}\\Desktop\\Docs\\Microsoft\\Excel",
140. '.xlr': f"{UserProfile}\\Desktop\\Docs\\Microsoft\\Excel",
141. '.xls': f"{UserProfile}\\Desktop\\Docs\\Microsoft\\Excel",
142. '.xlsx': f"{UserProfile}\\Desktop\\Docs\\Microsoft\\Excel",
143. # System
144. '.bak': f"{UserProfile}\\Desktop\\Other\\System",
145. '.cab': f"{UserProfile}\\Desktop\\Other\\System",
146. '.cfg': f"{UserProfile}\\Desktop\\Other\\System",
147. '.cpl': f"{UserProfile}\\Desktop\\Other\\System",
148. '.cur': f"{UserProfile}\\Desktop\\Other\\System",
149. '.dll': f"{UserProfile}\\Desktop\\Other\\System",
150. '.dmp': f"{UserProfile}\\Desktop\\Other\\System",
151. '.drv': f"{UserProfile}\\Desktop\\Other\\System",
152. '.icns': f"{UserProfile}\\Desktop\\Other\\System",
153. '.ini': f"{UserProfile}\\Desktop\\Other\\System",
154. '.lnk': f"{UserProfile}\\Desktop\\Other\\System",
155. '.msi': f"{UserProfile}\\Desktop\\Other\\System",
156. '.sys': f"{UserProfile}\\Desktop\\Other\\System",
157. '.tmp': f"{UserProfile}\\Desktop\\Other\\System",
159. '': f"{UserProfile}\\Desktop\\Folders"
160. }

163. def Cleaner():
164. for filename in os.listdir(folder\_to\_track):
165. i = 1
166. if filename not in ['desktop.ini', 'automations']:
167. try:
168. new\_name = filename
169. extension = 'noname'
170. try:
171. extension = str(os.path.splitext(
172. folder\_to\_track + '/' + filename)[1])
173. except Exception:
174. extension = 'noname'
176. folder\_destination\_path = extensions\_folders[extension]
177. if not os.path.exists(folder\_destination\_path):
178. os.makedirs(folder\_destination\_path)
180. file\_exists = os.path.isfile(
181. folder\_destination\_path + "/" + new\_name)
182. while file\_exists:
183. i += 1
184. new\_name = os.path.splitext(folder\_to\_track + '/' + filename)[0] + str(
185. i) + os.path.splitext(folder\_to\_track + '/' + filename)[1]
186. new\_name = new\_name.split("/")[4]
187. file\_exists = os.path.isfile(
188. folder\_destination\_path + "/" + new\_name)
189. src = folder\_to\_track + "/" + filename
191. new\_name = folder\_destination\_path + "/" + new\_name
192. os.rename(src, new\_name)
193. except Exception as e:
194. print(e)

197. Cleaner()

7.7Create project(Flutter)

1. import argparse
2. import os
3. from github import Github

6. def local():
7. try:
8. if os.path.exists(path):
9. os.chdir(path)
10. else:
11. print('Cannot create a file when that file already exists')
12. exit()
13. os.system(f'flutter create {foldername}')
14. os.chdir(\_dir)
15. os.system('git init')
16. os.system(f'echo "# {foldername}" > README.md')
17. os.system('git add README.md')
18. os.system('git commit -m "first commit"')
20. print(f'{foldername} created locally')
21. os.system('code .')
23. except Exception as e:
24. print(e)

27. def remote():
28. try:
29. g = Github(token)
30. user = g.get\_user()
31. login = user.login
32. user.create\_repo(foldername)
33. except Exception as e:
34. print(e)
36. commands = ['git init',
37. f'git remote add origin https://github.com/{login}/{foldername}.git',
38. 'git add .',
39. 'git commit -m "Initial commit"',
40. 'git push -u origin master']
42. try:
43. os.chdir(path)
44. os.system(f'flutter create {foldername}')
45. os.chdir(\_dir)
46. for c in commands:
47. os.system(c)
49. print(f'{foldername} initialized')
50. os.system('code .')
52. except Exception as e:
53. print(e)

56. if \_\_name\_\_ == "\_\_main\_\_":
58. UserProfile = os.environ.get('USERPROFILE')
60. # add projects dirctory to the env vars
61. path = f'{UserProfile}\\Desktop'
62. token = os.environ.get('gt') # add github token to the env vars
64. p = argparse.ArgumentParser()
65. p.add\_argument('name', help='name of the project')
66. p.add\_argument('-f', default='l', help='name of the project')
67. args = p.parse\_args()
69. foldername = args.name
70. \_dir = path + '\\' + foldername
72. if args.f == 'l':
73. local()
74. elif args.f == 'r':
75. remote()
76. else:
77. print('fpia --help for help')

7.8 Main\_start.cmd

1. @echo off
3. start cmd /k launch.cmd
5. start cmd /k ngrok-start.cmd
6. timeout /t 5 /nobreak
7. python PostingLocation.py
9. start https://alexautomation.herokuapp.com/read?id=url
10. start https://alexautomation.herokuapp.com/

7.9Results.py

1. import re
2. from bs4 import BeautifulSoup
3. import requests
4. import json
5. import httplib2
6. import urllib.request
7. from selenium import webdriver
8. import argparse
10. url = 'https://jntuaresults.ac.in/'
12. try:
13. src = requests.get('https://jntuaresults.ac.in/').text
14. except:
15. print('no internet')
16. exit()
18. soup = BeautifulSoup(src, 'lxml')
20. table = soup.find('table', {'class': 'ui table segment'})

23. def getRes\_a(value1, value2):
24. geturl = ""
25. year=value2.split(" ")[1]
26. month=value2.split(" ")[0]
27. for row in soup.findAll("a", href=True):
28. if value1 in row.text:
29. # print(row.text)
30. h = row['href']
31. geturl = f'{url}{h}'
32. print(f'{row.text} : {geturl}')
33. if month in row.text:
34. if year in row.text:
35. return geturl
36. else:
37. return "result not found or has been deleted by Admin"



42. if \_\_name\_\_ == "\_\_main\_\_":
43. p = argparse.ArgumentParser()
44. p.add\_argument('val1', help='val 1')
45. p.add\_argument('val2', help='val 2')
46. args = p.parse\_args()
48. l=getRes\_a(args.val1, args.val2)
49. print(l)
50. if l!=None:
51. rollno = '163G1A0505'
52. browser = webdriver.Chrome(r"C:\Users\wikyprash\Desktop\Server\Executing\chromedriver.exe")
53. print(l)
54. browser.get(l)
55. field = browser.find\_element\_by\_xpath(
56. '/html/body/div/div[1]/div/div/center/table/tbody/tr/th/center/input[1]')
57. btn = browser.find\_element\_by\_xpath(
58. '/html/body/div/div[1]/div/div/center/table/tbody/tr/th/center/input[2]')
59. field.send\_keys(rollno)
60. btn.click()
61. else :
62. browser = webdriver.Chrome("E:\Final\_year\_projects\Client\_api\Executing\chromedriver.exe")
63. browser.get("https://alexautomation.herokuapp.com/")



68. '''
69. usage : python Rantest1.py "B.Tech IV Year I Semester (R15)" "Nov/Dec 2019"
71. '''

7.10 Loginer.py

1. from selenium import webdriver
2. from selenium.webdriver.support.ui import WebDriverWait
3. from selenium.webdriver.common.keys import Keys
4. from selenium.webdriver.support import expected\_conditions as EC
5. from selenium.webdriver.common.by import By
7. driver = webdriver.Chrome(r"C:\Users\wikyprash\Desktop\Server\Executing\chromedriver.exe")
8. def facebook():
9. WebDriverWait(driver, 10)
10. driver.get("https://facebook.com")
11. driver.find\_element\_by\_name("email").send\_keys("moory.ranjith@gmail.com")
12. driver.find\_element\_by\_name("pass").send\_keys("163g1a0548"+Keys.ENTER)
13. return "facebook as loged in"
14. facebook()

  7.11PostingLocation.py

1. import json
2. import os
3. import requests

6. def creating\_url():
7. try:
8. os.system("curl http://localhost:4040/api/tunnels > tunnels.json")
9. with open('tunnels.json') as data\_file:
10. dataJson = json.load(data\_file)
11. return dataJson
12. except Exception as e:
13. print(e.args)
14. print("start ngrok to get tunnel the localhost")

17. def posting\_URL(\_url):
18. \_main\_url = "https://alexautomation.herokuapp.com/add?url="
19. try:
20. respone = requests.post(\_main\_url+\_url)
21. return print(respone.text)
22. except expression as identifier:
23. print(identifier)

26. if \_\_name\_\_ == "\_\_main\_\_":
27. url = ""
28. jsonData = creating\_url()
29. for i in jsonData['tunnels']:
30. url = url+i['public\_url']+'\n'
31. print(posting\_URL(url.split("\n")[0]))

**8. SYSTEM TESTING**

**8.1:** **INTRODUCTION:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2: TYPES OF TESTS:**

**Unit testing:**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test:**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**SYSTEM TESTING:**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**WHITE BOX TESTING:**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**BLACK BOX TESTING:**

Black Box Testing is testing the software without any knowledge of the innerworkings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**UNIT TESTING:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach:**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives:**

* User must Invocation the Alexa skill.
* User devices has to send it location to server.
* The entry screen, messages and responses must not be delayed.

**Features to be tested:**

* Verify that the command is of the correct format
* Checking that commands has passed to AWS lambda
* Request form server has to execute in user device.

**INTEGRATION TESTING:**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

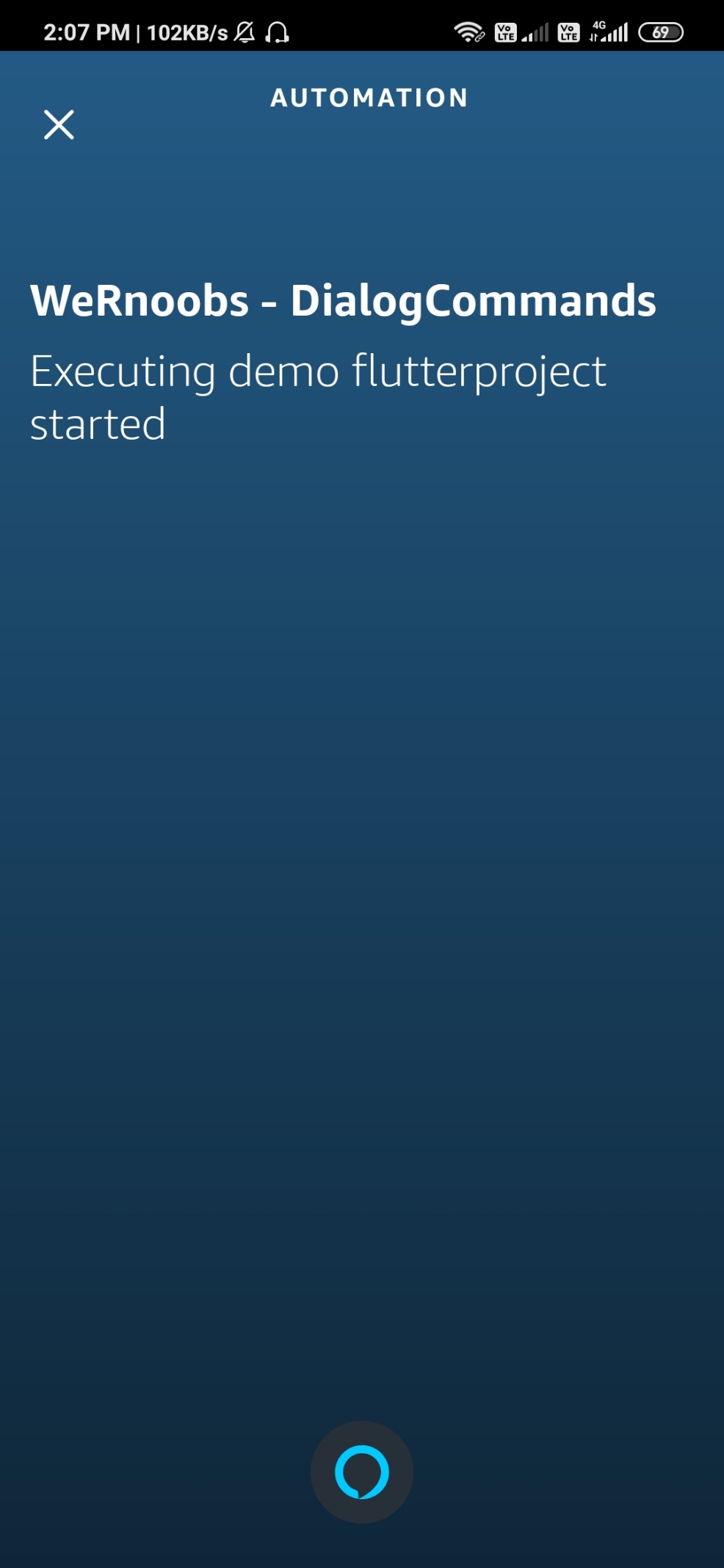
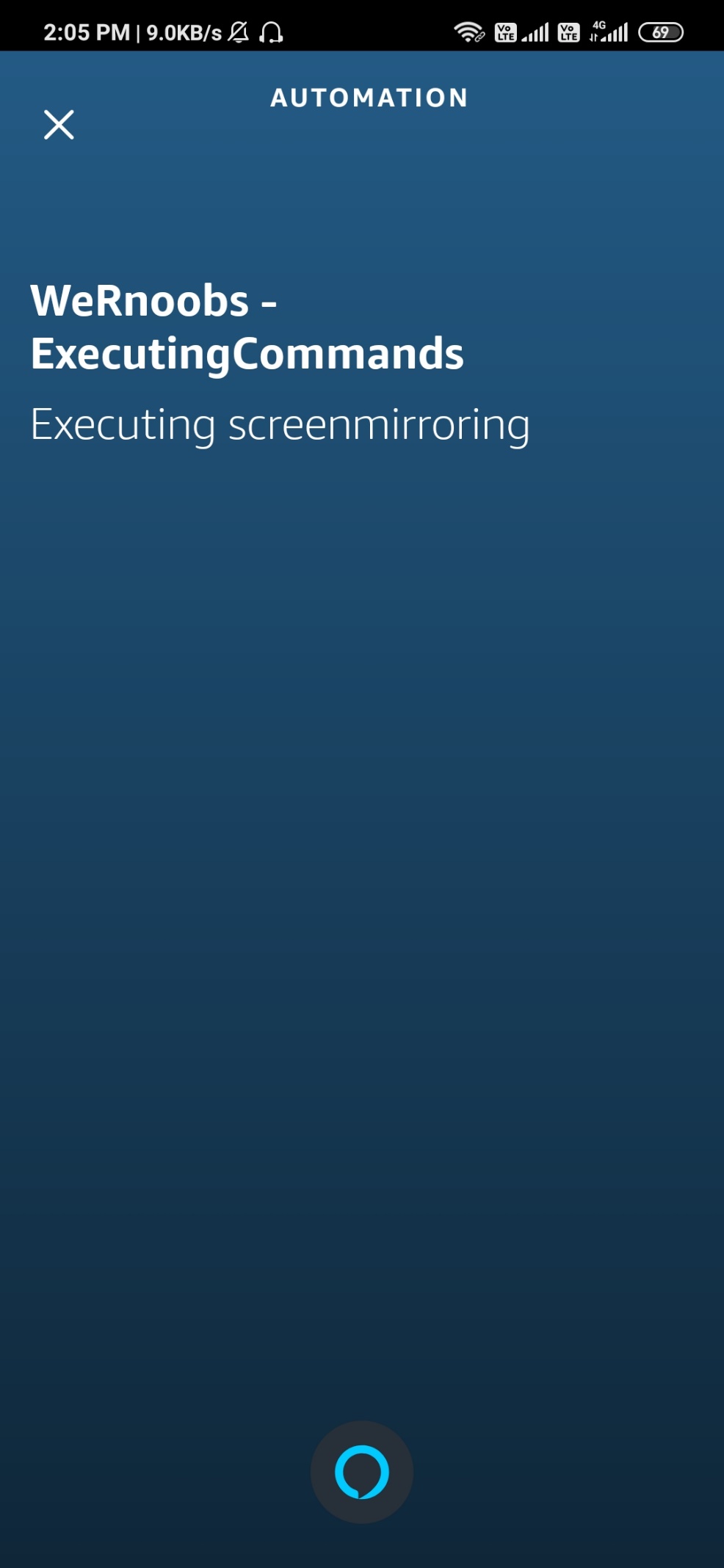
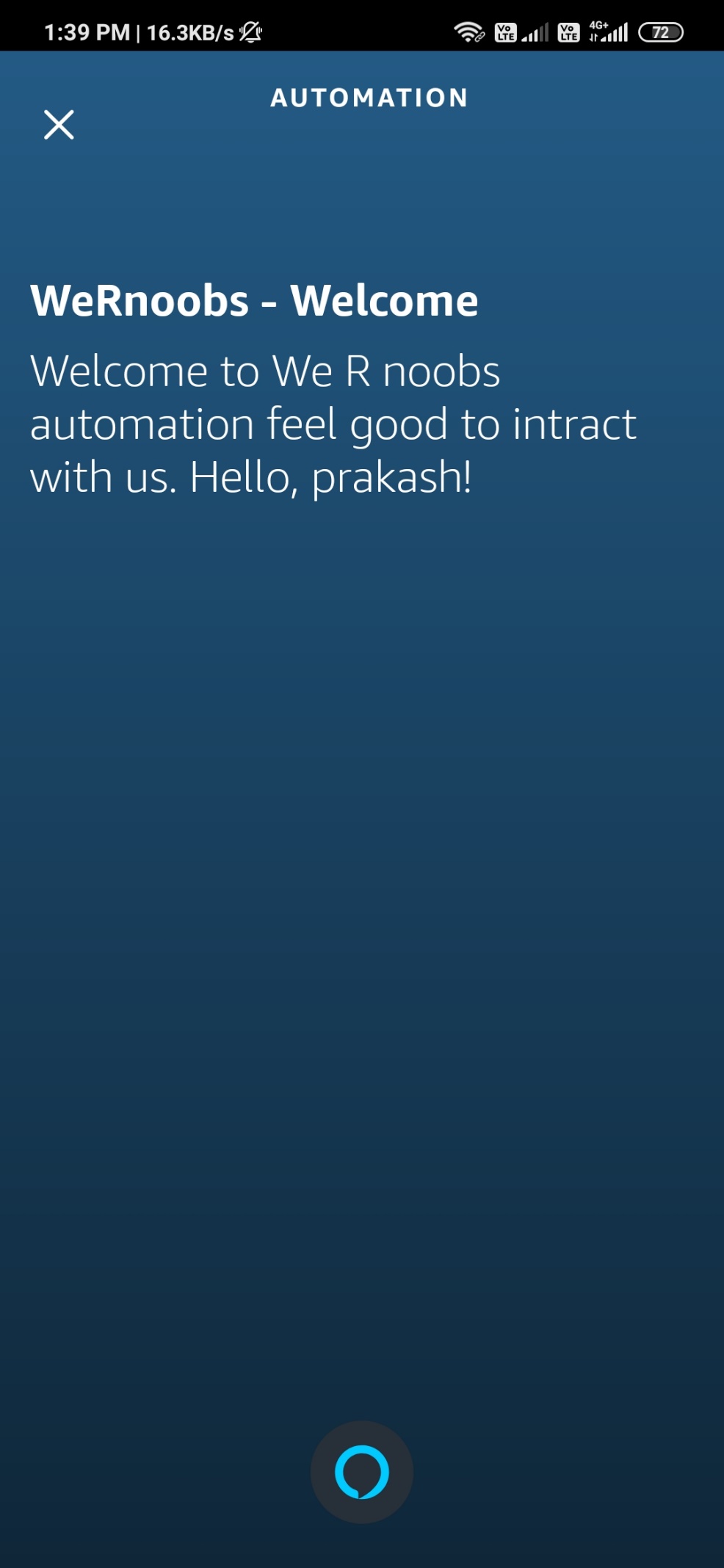
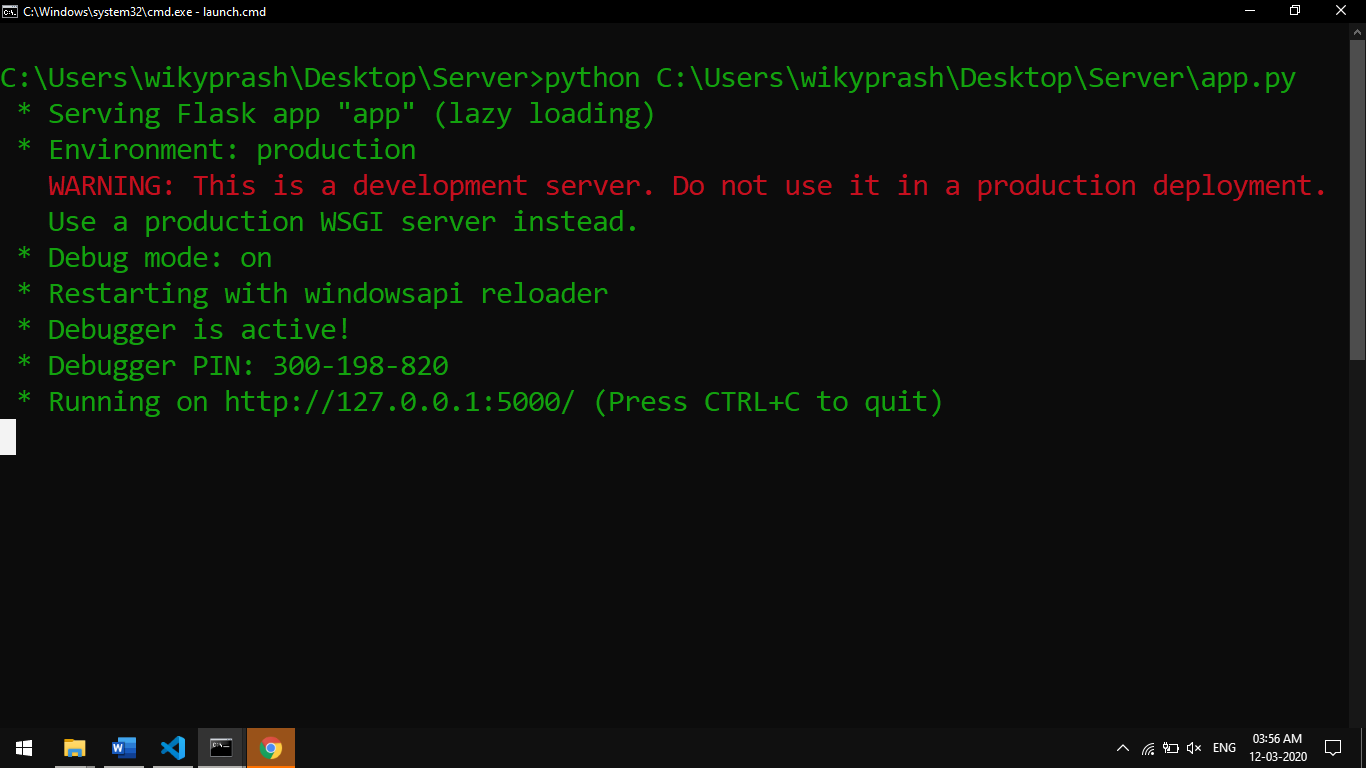
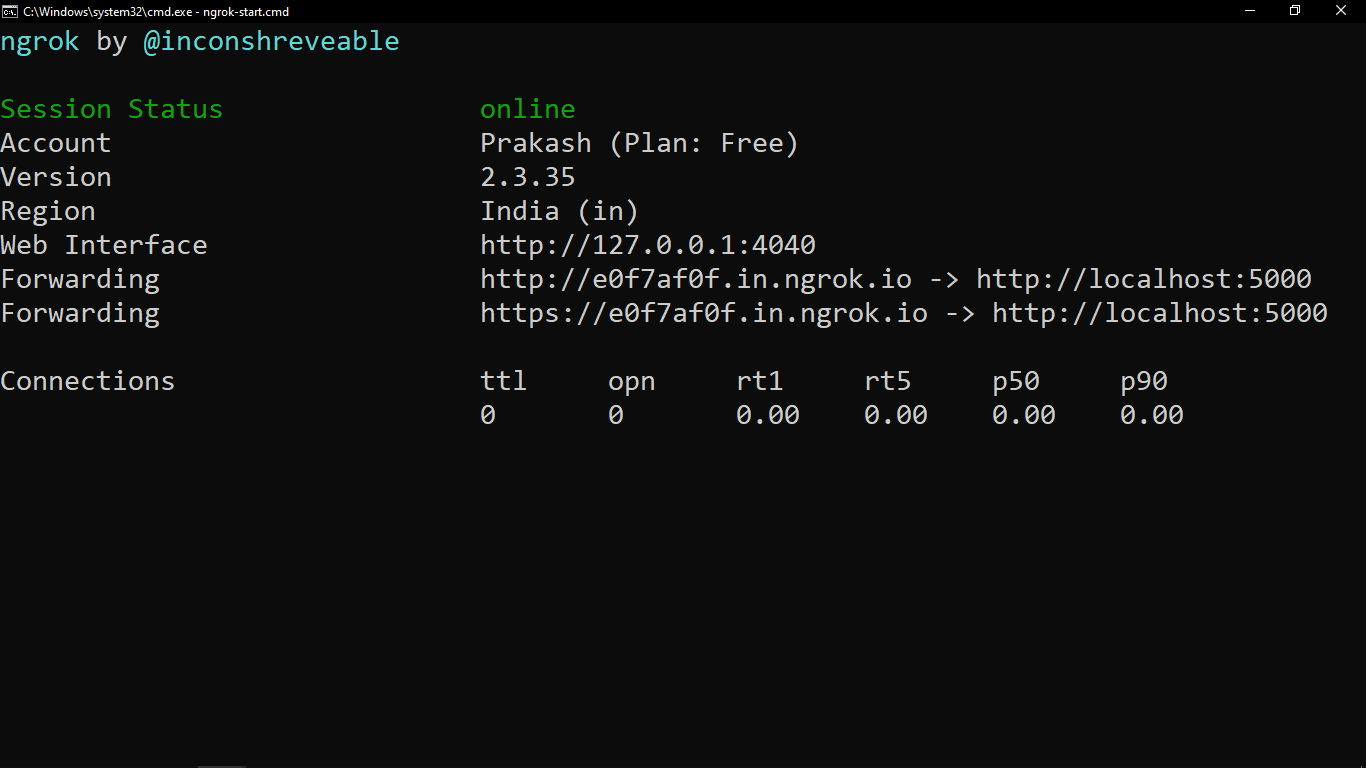
**Test Results:** All the test cases mentioned above passed successfully. No defects encountered. **ACCEPTANCE TESTING*:***

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountere

**9. OUTPUT SCREENSHOTS**

**Servers**



**10. CONCLUSION**

In this paper,