

Frugal Testing STE Assignment Submission

- Candidate Name: Vajrapu Navyasree
- Name: Vishnu Institute of Technology
- Roll Number: 22PA1A12I4

SECTION 0: Mandatory Questions

1. Please confirm your consent to sign a bond for 30 months (including a 1-year internship). Only

Yes, I fully consent to signing a bond with Frugal Testing for a period of 30 months, including the 1-year internship. I understand that the company is investing its time, resources, and mentorship in my growth, and I take this seriously. A long-term commitment like this allows me to fully immerse myself in the company's learning environment and deliver meaningful contributions. I also believe that sticking to a commitment reflects reliability and professionalism—qualities I want to develop further. I have no hesitation in accepting this term, as I see it as an opportunity to grow technically and professionally.

2. Submit your assignment if you are comfortable with this bond.

Yes, I am completely comfortable with the bond and therefore submitting this assignment with a clear understanding of its implications. I respect that Frugal Testing expects dedication, discipline, and consistency during the 30-month period. To me, this is a fair ask, especially when the company offers structured training, real-time project exposure, and a future PPO. I am confident that this journey will add immense value to my career. I see this not just as a bond but as a commitment to mutual growth, where I will give my best and receive practical learning and long-term development in return.

3. Have you received the stipend and CTC details? Please confirm by providing the details here.. Are you willing to relocate to Gachibowli, Hyderabad?

Yes, I have received the stipend and CTC details shared by Frugal Testing. During the 1-year internship period, the monthly stipend will be ₹15,000,

which is a generous and motivating amount for a fresher. After successful completion of the internship, the company offers a Pre-Placement Offer (PPO) with a CTC of ₹5 LPA. Additionally, there is a standard notice period of 3 months, and the possibility of up to 10% appraisal based on performance and behavior. I find this offer both fair and competitive, and I truly appreciate the transparency and structure the company has shared. Yes, I am willing and prepared to relocate to Gachibowli, Hyderabad to work from Frugal Testing's office. I believe that early career roles are most effective when done on-site, especially in a collaborative environment. Being physically present at the office will help me communicate better with my mentors and teammates, understand the work culture, and improve faster through direct feedback. I also see relocation as a step forward in adapting to the industry. I have already discussed this with my family and am arranging accommodation, so I can move as soon as required and start contributing immediately.

4. What motivated you to pursue a career in software testing?

My motivation to pursue a career in software testing began when I realized how critical it is to ensure software works correctly before reaching users. During my college projects, I often found myself focusing more on checking if everything was working as expected, identifying bugs, and validating results. This experience made me understand that testing is not just about finding defects—it's about ensuring the best possible user experience. I enjoy working through logic, writing test cases, and using tools like Selenium to automate repetitive tests. Testing allows me to blend technical skills with analytical thinking, which aligns with my strengths.

5. Why do you want to join a software testing firm like Frugal Testing?

Frugal Testing stands out to me because it is specialized in software testing and DevOps—two of the most in-demand and fast-evolving domains in the tech industry. As someone who wants to build a solid foundation in testing, I believe being part of a niche firm like Frugal will give me more focused learning and real-world exposure. Their structured internship, emphasis on discipline and mentorship, and hands-on client projects make it an ideal environment for growth. Also, the company's transparent approach and

strong emphasis on quality match the kind of work culture I'm looking to start my career in.

SECTION A:

2. Automate Job Search and Application on Shine Jobs

Demo Url:

<https://drive.google.com/file/d/1CN6nlSbgPKbVgBvaQEMTYms25PyBh2Ys/view?usp=sharing>

Screenshots and Logs:

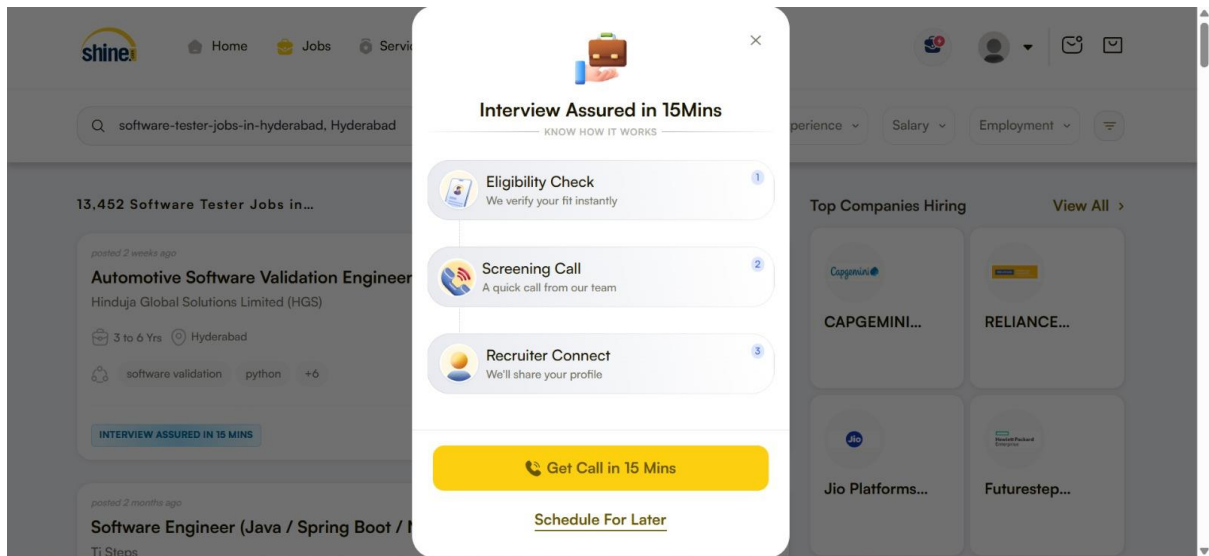
The image displays two screenshots of the Shine Jobs website interface.

Top Screenshot: User Profile Dashboard

- Navigation Bar:** Includes links for My Shine, Search Jobs, Jobs for You, Mailbox, Profile, My Job Alerts, Services, Career Guidance, and a user profile icon.
- Search Filters:** Job title, skills (Software Tester), Location (Hyderabad), Experience (2).
- Search Button:** Search jobs.
- Dashboard Metrics:** 0 Appeared to Recruiters, 1 Recruiter Actions, 0 Messages, 0 Job alerts.
- Recommended Jobs:** A section showing three job listings with details like job title, company, location, and experience requirements. Each listing has a button to 'Apply'.
- Right Sidebar:** Contains a 'Career Guidance' section with a 'Update Profile' button and a 'Empower Skills, Elevate Careers' section with a 'View Upcoming Workshops' button.

Bottom Screenshot: Search Results

- Search Bar:** Contains the search query 'software-tester-jobs-in-hyderabad, Hyderabad'.
- Filters:** Sort, Location, Experience, Salary, Employment.
- Results:** 13,452 Software Tester Jobs in... Toggle to save search.
- Top Companies Hiring:** A list of companies including Capgemini, Reliance, Jio Platforms, and Futurestep.
- Job Listing:** A detailed view of a job listing for 'Automotive Software Validation Engineer' at Hinduja Global Solutions Limited (HGS), located in Hyderabad, with a salary range of 3 to 6 Yrs. The listing includes a 'Apply' button.



LOGS:

```

16 def main():
17     top_login_btn = wait.until(EC.element_to_be_clickable(
18         (By.XPATH, "//button[contains(text(),'Login')]"))
19     )
20     top_login_btn.click()
21
22     # Step 3: Enter credentials
23     email_field = wait.until(EC.visibility_of_element_located((By.ID, "id_email_login")))
24     email_field.send_keys("navyasri11488@gmail.com")
25
26     password_field = driver.find_element(By.ID, "id_password")
27     password_field.send_keys("Navyasri@123")
28
29     final_login_btn = driver.find_element(
30         By.XPATH,
31         "//*[@id='candidate_login_widget']/div[1]/form[5]/ul[1]/li[4]/div/button"
32     )
33     final_login_btn.click()
34
35
36
37
38
39
40
41
42
43
44
45
46
47

```

```

[12188:2164:1003/123442.315:ERROR:google_apis\gcm\engine\registration_request.cc:291] Registration response error message: DEPRECATED_ENDPOINT
Job Title: Software Engineer (Java / Spring Boot / Node.js)
Company Name: Ti Steps
Applied successfully!
PS C:\Users\Asus\Desktop\Testing>

```

SECTION B:

3. Case Study and Testing Challenges

1. **Case Study Chosen:** Performance Testing for a Fin-Tech Company in Singapore.

Testing Challenge Identified: The client, a leading fintech company, needed their web platform to handle over **20,000+ concurrent users** without any drop in performance. The key challenge was ensuring that the **response time stayed under 1 second**, even during peak loads. Since users relied on real-time stock

information to make investment decisions, any delay in page loads or data retrieval could result in a poor user experience or even financial loss. Another critical concern was scalability—the system had to scale quickly and efficiently as user traffic fluctuated. Their backend database, server configuration, and overall infrastructure needed optimization to meet these high-performance standards.

Solution Implemented:

Frugal Testing began with a complete **vulnerability and architecture review** of the platform. They conducted **performance benchmarking** to test the infrastructure's ability to handle large volumes of simultaneous requests. A detailed review of the **MySQL database** helped identify slow queries and I/O bottlenecks. Simple yet impactful code changes were made to optimize response times. They also implemented auto-scaling at the **server level**, which allowed the system to expand or shrink based on user load. The team used tools to monitor **thread handling, memory usage, and connection pools** to make sure nothing was being over-utilized or under-managed.

If I were assigned to this project:

If I were assigned to this project, I would use tools like **Apache JMeter** or **K6** to simulate high user loads and analyze system performance. For real-time monitoring, I'd integrate **Grafana with Prometheus** to track CPU, memory, and latency metrics. I'd also optimize slow SQL queries using **MySQL Query Profiler** and conduct stress testing during peak and off-peak times. To improve global response times, I would implement a **CDN** like Cloudflare. My focus would be on identifying bottlenecks early and applying performance tuning strategies to ensure scalability and a seamless user experience.

2. Blog Chosen: "Why Shift-Left Testing is the Future of QA"

New Insights Gained:

- **Shift-left testing means** moving testing activities earlier into the development cycle—starting from the requirement phase.

- I learned that this helps catch bugs before the code is even written, reducing cost and improving turnaround time.
- Frugal Testing's blog emphasized the role of **BDD (Behavior-Driven Development) and unit-level test automation** in effective shift-left testing.
- Another key insight was that early testing improves team collaboration as QA, developers, and business analysts work together from day one.

4. Real-World Testing Challenges

In my complaint management system project, I encountered an issue where certain complaints were not being saved in the database. After reviewing the code, I found that some form fields were not properly validated, and missing values caused the data insertion to fail silently. Initially, this bug was overlooked during testing because we focused only on successful submissions. I resolved it by adding required field validation on both frontend and backend. This experience taught me the importance of handling edge cases and validating user input thoroughly in testing.

5. Collaboration & Teamwork

A. When working with team members who had different communication styles during the FinGenius project, I adapted based on their preferences. I used clear documentation for those who preferred written communication and quick calls for real-time collaboration. This flexibility helped maintain clarity and avoid misunderstandings.

B. In the FinGenius project, I worked with a diverse team including developers and content writers. We faced coordination issues initially, so I proposed structured updates and task tracking. This improved communication and helped us deliver the project smoothly. We also created a shared workspace to keep everyone aligned. The experience taught me the value of proactive collaboration and taking initiative when team dynamics need support.

6. Applying Technical Skills to Achieve Success

In my complaint management project, I used my skills in Node.js, Express, and MongoDB to build a secure and functional backend system. I implemented authentication, role-based access, and proper form validation. Initially, I faced issues with managing errors and routing due to limited experience, but I overcame them through hands-on debugging and learning from documentation. These efforts ensured smooth complaint tracking and submission. To improve further, I'm strengthening my frontend development skills and learning better ways to test and structure full-stack applications.

7. Essential Tools in Software Testing

Common testing tools include **Postman**, **JMeter**, **Selenium**, **TestRail**, **Bugzilla**, **TestLink**, **Cypress**, and **Playwright**.

Two tools I find especially useful are:

- **Postman**: It helps testers validate APIs by sending requests, verifying responses, and automating test collections for backend services.
- **TestRail**: It allows structured test case management, making it easy to track testing progress, assign tasks, and generate test reports.

These tools improve accuracy, speed, and transparency in both manual and automated testing workflows.

8. Career Growth & Learning in Software Testing

I see myself growing into a QA Engineer role with expertise in both functional and automation testing. Over time, I want to lead test planning and strategy for real-world projects. To stay updated, I follow software testing blogs, take online courses, and practice hands-on with real tools. I also explore open-source projects, participate in communities, and keep track of trends like low-code automation, API testing, and testing in CI/CD pipelines to stay ahead in the field.

9. Impact of DevOps & Cloud Computing on Testing

DevOps practices integrate testing into development through continuous integration and delivery (CI/CD), which speeds up releases and ensures better quality. Testing becomes more collaborative and automated. **Cloud testing** helps simulate real-world scenarios across devices and locations, offering scalability and cost savings. However, challenges include data security and managing shared environments. **CI/CD pipelines** help improve quality by running automated tests after every code change, catching bugs early, and reducing manual intervention in the deployment cycle.

10. Profile Details

LinkedIn: <https://www.linkedin.com/in/vajrapu-navya-sree-a05561276/>

Resume: <https://drive.google.com/file/d/1ZPauKvOgy7iI0LQCLCytgF8lvqMD-ruv/view?usp=sharing>

GitHub: <https://github.com/navyasri124>

Technical Profiles:

1.LeetCode: <https://leetcode.com/problemset/>

2.GeeksForGeeks: <https://www.geeksforgeeks.org/user/navyasryxp9/>

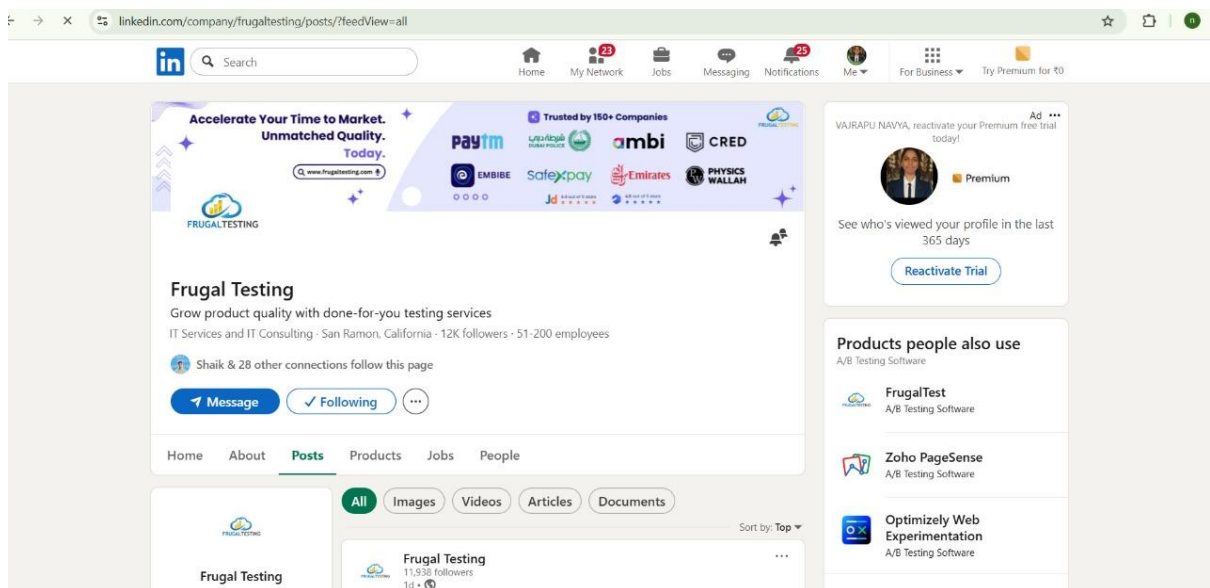
Project Link:

1. <https://github.com/navyasri124/Vit-Complaints>

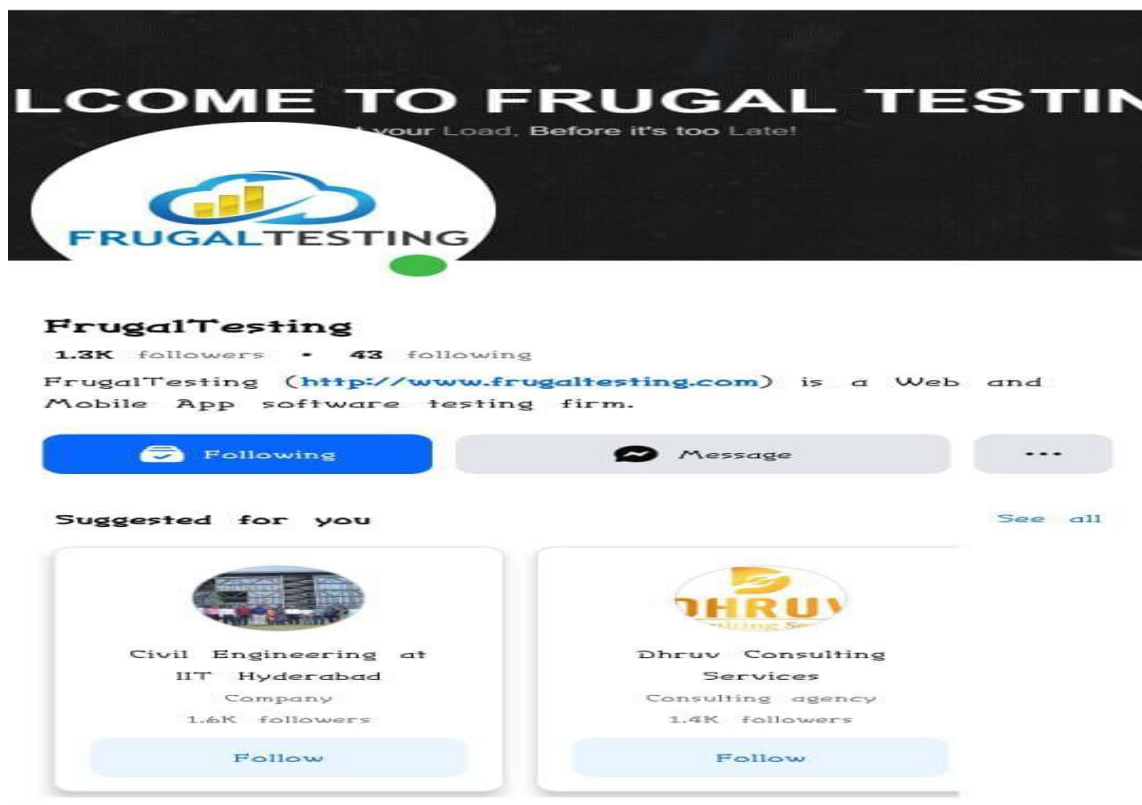
2. <https://github.com/navyasri124/Educational-Organization-using-servicenow-project>

11. Social Media Task:

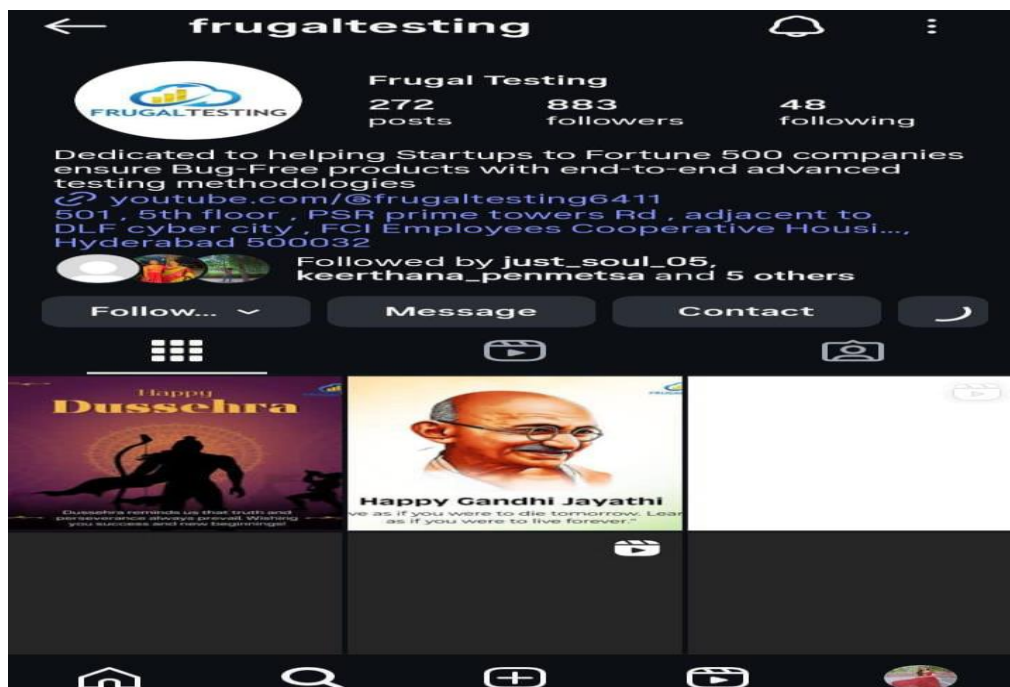
LinkedIn:



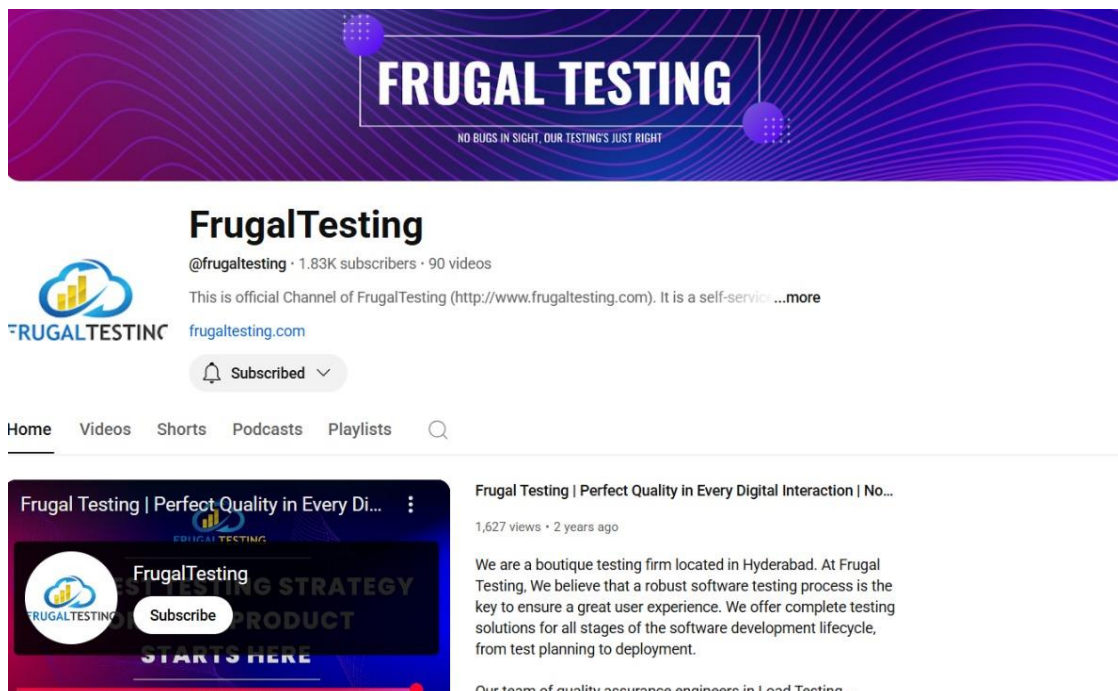
Facebook:



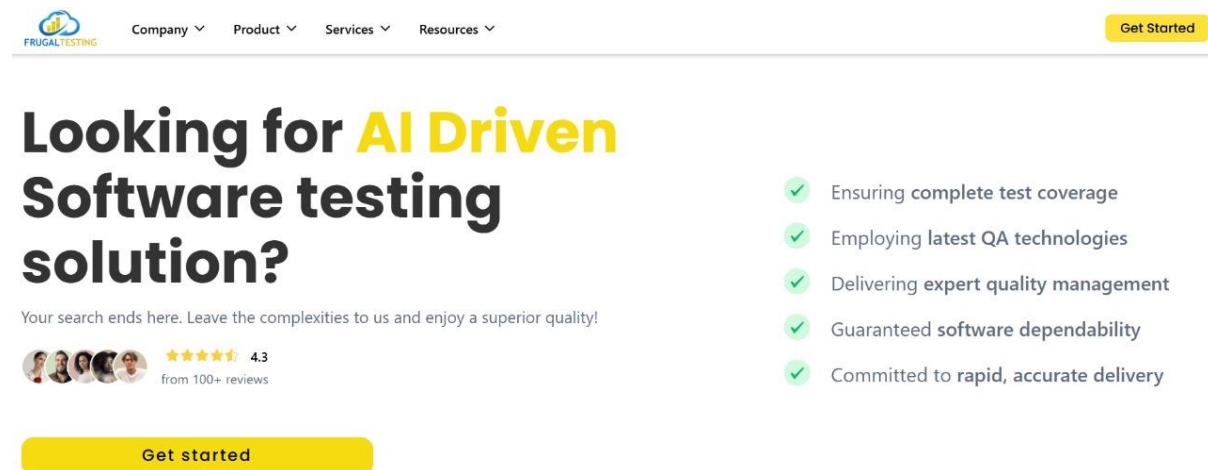
Instagram:



YouTube:



Google Page:



The image shows the top portion of the Frugal Testing website. At the top is a navigation bar with the Frugal Testing logo on the left and links for Company, Product, Services, and Resources in the center. A yellow 'Get Started' button is on the right. Below the navigation bar is a large hero section with the heading 'Looking for AI Driven Software testing solution?' in bold black and yellow text. Underneath the heading is a sub-headline: 'Your search ends here. Leave the complexities to us and enjoy a superior quality!'. To the left of the sub-headline is a row of five profile pictures and a 4.3 star rating from 100+ reviews. To the right of the sub-headline is a list of five benefits, each preceded by a green checkmark: 'Ensuring complete test coverage', 'Employing latest QA technologies', 'Delivering expert quality management', 'Guaranteed software dependability', and 'Committed to rapid, accurate delivery'. At the bottom of the hero section is a yellow 'Get started' button.

FRUGAL TESTING

Company Product Services Resources

Get Started

Looking for AI Driven Software testing solution?

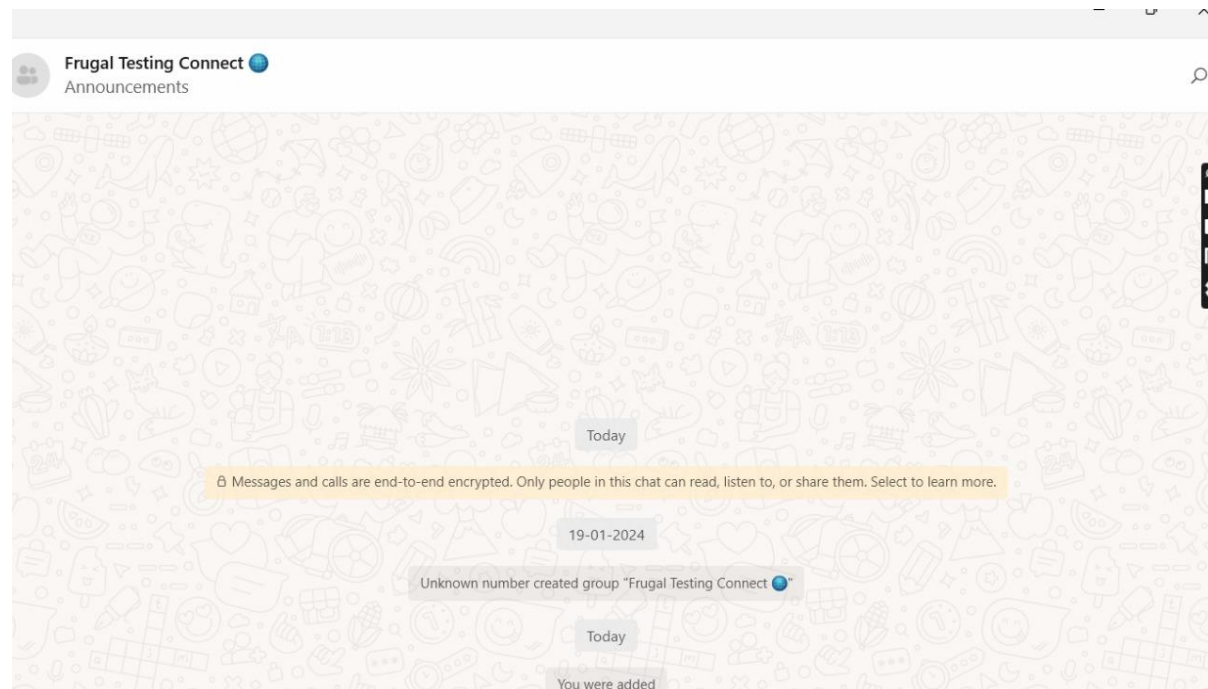
Your search ends here. Leave the complexities to us and enjoy a superior quality!

★★★★★ 4.3
from 100+ reviews

- ✓ Ensuring complete test coverage
- ✓ Employing latest QA technologies
- ✓ Delivering expert quality management
- ✓ Guaranteed software dependability
- ✓ Committed to rapid, accurate delivery

Get started

WhatsApp Community:



12. Understanding the Bug Lifecycle

The **bug lifecycle** (or defect lifecycle) refers to the journey of a bug from discovery to closure. The stages include:

- New
- Assigned
- Open
- Fixed
- Retest
- Verified

- **Closed**
- **Reopened (if needed)**

Key factors influencing the status are severity, reproducibility, developer response, test results, and timelines. A clear workflow helps teams track progress and ensures no bugs are missed before release.

13. Closing a Bug

A bug is marked “**Closed**” after it’s been fixed, retested by QA, and confirmed to work as expected.

Steps before closing:

- The bug fix is verified in the test environment.
- All test cases related to the bug are passed.
- No regression or side effects are introduced.
- Relevant stakeholders (dev + QA) agree on the resolution. Only after these validations, the tester marks it as closed in the tracking system.

14. Leveraging AI in Software Testing

AI-based tools can optimize testing by identifying repetitive tasks, generating test cases, and auto-detecting UI changes.

For example, **AI can analyze past test runs** to suggest high-risk areas or even create smart test suites.

Advantages:

- Faster test execution
- Smarter defect detection
- Improved coverage
- Risks:
 - May miss context-specific bugs
 - Over-dependence can reduce human analysis So, AI should assist testers—not replace them

Risks:

- May miss context-specific bugs
- Over-dependence can reduce human analysis So, AI should assist testers—not replace them.

15. Severity vs. Priority in Bug Management

Severity- It refers to the technical impact a bug has on the system's functionality

Priority- It defines how urgently the bug needs to be addressed based on business needs.

Example 1 – High Severity, Low Priority:

A crash occurs when accessing a rarely used feature on an outdated OS version. It's a major functionality issue (high severity), but fixing it isn't urgent due to limited user impact (low priority).

Example 2 – Low Severity, High Priority:

A typo in the company name on the landing page. It doesn't affect functionality (low severity) but needs immediate correction for brand reputation (high priority). Accurate severity and priority assignment helps teams triage bugs efficiently and focus on what matters most in a release cycle.

16. ChatGPT and AI Tools in Testing

Tools like ChatGPT can assist testers by generating sample test cases, writing boilerplate code for automation scripts, or explaining error logs in simple terms. It can also help create test data quickly, saving time during early testing phases. However, AI tools have limitations—they lack real-time application context and may provide inaccurate suggestions if the input isn't clear. They can't replace manual exploratory testing or critical thinking. While ChatGPT boosts productivity, it's best used as a support tool to assist testers—not as a replacement for hands-on testing experience and human judgment.

17. Regression Testing for Non-Technical Stakeholders

Regression testing is done to make sure that existing features still work correctly after new code changes are made. For example, imagine we fix a bug in the login system—regression testing ensures that this fix doesn't accidentally break the signup, dashboard, or logout features. It's like updating one part of a machine and then checking if the whole machine still works properly. This testing is critical to maintain software stability, especially when frequent updates are made during development.

18. Deciding Between Automation & Manual Testing

The decision to automate or manually test a feature depends on factors like test frequency, stability of the feature, complexity, and time constraints. Automation is ideal for repetitive tasks like login validations, form submissions, and regression tests—where the test steps don't change often. Manual testing is better when the UI is frequently changing or when testing

needs human observation, such as in usability or exploratory testing. For example, testing the look and feel of a new user interface is better done manually, while repeated data validation can be automated to save time and effort.

19. Different Testing Types

Sanity Testing is a quick check to ensure that specific features or bug fixes work as expected before deeper testing begins. For example, after fixing the complaint submission bug, we check if the form now submits correctly.

Smoke Testing verifies that the major functionalities of an application are working after a new build. For instance, checking if login, dashboard, and logout work in a fresh deployment.

Regression Testing ensures that new changes haven't broken existing features. For example, after adding a new status update feature in a complaint app, we test if old features like viewing and filtering complaints still work.

Monkey Testing is a form of random, unplanned testing to check the system's stability. It's like clicking buttons or entering random inputs to see if the app crashes unexpectedly.

20. The Evolution of Software Testing: Past, Present, and Future

Introduction

Software testing has always been an important part of software development, but its role has changed a lot over time. In the early days, testing was done manually and only at the end of a project. Today, it starts early, runs continuously, and involves both manual and automated methods. With new technologies like AI and cloud computing, the future of testing looks even more advanced. This article explains how software testing has evolved—from its early days to its modern practices, and what we can expect in the years to come.

The Past

In the early stages of software development, testing was mostly manual and done at the final phase of the project. Developers would complete the entire code, and only then would testers check if it worked properly. This process was slow and risky. If a bug was found near the end, it could delay the project or even lead to failure. Testing was not seen as important in the beginning, and testers had very limited tools—most used spreadsheets or written documents to keep track of test cases.

The communication between testers and developers was also weak. Testers worked separately, and developers often didn't get feedback on time. Testing was done following the Waterfall model, where each phase of development came one after another. Because of this, testing was seen as a final check, not something that helped improve the software throughout development. Still, this period helped create basic testing processes like writing test plans, tracking bugs, and reporting issues in a structured way.

The Present

Today, software testing has become much more advanced and efficient. In modern development, especially with Agile and DevOps, testing is done throughout the entire software development cycle. Testers now work closely with developers, often in the same team, and take part in planning and reviewing features from the beginning. This is known as "Shift-Left Testing" because it moves testing to the earlier part of the process.

One of the biggest improvements is the use of automated testing. Tools like Selenium, Postman, and Cypress allow testers to write scripts that automatically check whether features are working properly. This saves a lot of time and is very useful for repetitive tasks, like testing login forms or checking if a page loads correctly. Testers also use Continuous Integration/Continuous Deployment (CI/CD) tools like Jenkins or GitHub Actions to run tests automatically whenever new code is added.

Modern testing also includes checking how software behaves on different devices, browsers, or networks using tools like BrowserStack and Appium. Testers don't just find bugs anymore—they help improve user experience, performance, and even suggest better ways to build the software. The role has become more technical, and testers are now often called "Quality Engineers."

The Future

The future of software testing will be shaped by technologies like Artificial Intelligence (AI), Machine Learning (ML), cloud computing, and Internet of Things (IoT). As software becomes more complex, testers will need smarter tools to keep up. AI can help by automatically creating test cases, identifying risky areas, and analyzing huge amounts of test results faster than humans can.

Some modern tools already offer "self-healing tests," which can adjust themselves when a part of the application changes, reducing the need for manual updates. We may also see more tools where testers can write test cases in plain English, and the system will convert them into executable scripts. This will make automation easier for everyone, even for those who don't have strong coding skills.

In the future, testing will also cover new areas like voice apps, smart devices, wearables, and even augmented reality. Testers will need to learn about these platforms and find new ways to check that they work correctly. As a result, testing will become more creative and more focused on how real users interact with software.

Conclusion

Software testing has evolved from being a slow, manual task done at the end of development to becoming a fast, continuous, and smart process that runs throughout the project. In the past, it was mostly about finding bugs. Today, it's about improving the overall quality of the product. In the future, it will become even more intelligent and automated with the help of AI and machine learning. For anyone entering the field of software testing, it's important to understand this evolution. Learning both manual and automated testing, working with modern tools, and staying updated with new technologies will be key to building a successful

career in quality assurance. As software keeps evolving, testing will continue to play a central role in delivering great user experiences.