# TABLE OF CONTENTS

# 1 INTRODUCTION

This test plan document documents specifies the strategy being followed for Testing of the project and sample of the test cases being written for the project

# 2 TESTING MODEL

During the development phase, we used the prevention model as our primary model for testing. The reason for using this was to prevent errors in the error by including all kinds of use-cases and corresponding error codes. Each of the module designed was planned accordingly and hence the prevention model was the most appropriate model for testing.

# 3 TESTING TYPES BEING USED

## 3.1 List all the Different Types of tests WHICH you plan to run

1. **Unit Testing:**
    a. Unit testing was done on individual components developed for any syntactic or semantic errors found in the code. Boundary conditions were tested and corresponding output scenarios were noticed.
    b. To make the code efficient, continuous optimization were done. Optimization were in terms of the data structures used for fast retrieval of information, and the algorithm used for faster computation.
2. **Integration Testing:**
    a. Since each of the scrum teams worked on different features, the system had to be integrated in an incremental process bottom up approach. This was helpful since each component could be integrated step by step.
    b. Dependency between components were minimal since all the apis written were independent of each other.
    c. Continuous testing was done in order to make sure that the system is working as expected.
3. **Grey Box Testing:**
    a. In order to take the advantages or both black box testing as well as white box testing, grey box testing was employed.
    b. For testing the semantics of the algorithms and data structures that were used, white box testing was used and in order to test if the corresponding functionality works as expected, black box testing was used.
    c. Hence this combination led us to use grey box testing.

## 3.2 List the Phases of the LifeCycle and the V&V done for each phase in the project

1. **Requirement Engineering phase:**

a. During this phase, verification was done on whether any feature is being built in the right way.
b. Code review and walk through was done in order to check if a particular requirements is meeting its criteria.

2. **Design Phase:**
   a. During the design, all the related components were kept in one place in order to achieve high cohesion and low coupling.
   b. Corresponding code review was done to see if the design meets the requirements or not
3. **Code phase:**
   a. The decisions taken during the requirements and the design phase was adhered during the code phase and individual functionality were written in a modular approach.
   b. Since we are using django as our backend, each and every functionality were written in a class structure with different http methods being the functions inside the class.
   c. Appropriate HTTP response codes were returned for different use cases and code review and code walkthrough was done for static analysis of the code.
4. **Testing phase:**
   a. Appropriate testing models mentioned above were followed in order to test the software.

### 3.3 Set out the Test adequacy criteria for your project

1. The criteria that were set for this project are as follows:
   a. Appropriate functionalities were found for the software. If the functionalities are working as expected, then testing phase can be stopped.
   b. The statement coverage adequacy criterion: This is satisfied by a particular test suite for a particular program if each executable statement in the program (i.e., excluding comments and declarations) is executed by at least one test case in the test suite.
   c. A fault-based adequacy criterion that seeds a certain set of faults would be satisfied if, for each of the seeded faults, there is a test case that passes for the original program but fails for the program with (only) that seeded fault.

# 4 TEST CASES

## 4.1 TEST CASE 1

- Input validation for coordinator scheduling API
- Authenticated coordinator account
- Make a GET request with the company visit ID. The test cases are:

    - GET request with valid company visit ID
    - GET request with invalid company visit ID
- POSTMAN tool to be used as the testing environment
    - Result of test case 1: 200 (OK) Response Code
    - Result of test case 2: 400 (Bad Request) Response Code

### 4.2 TEST CASE 2

- Removing a booking made by the coordinator
- The coordinator must be authenticated and the company visit ID must be valid
- Click on the "Cancel Schedule" button and check the schedule for the company after that
- Test Environment/Data to be used
- Two possible scenarios:
    - The success scenario is when the schedule gets removed for that specific company on the database, and on checking the schedule, we should see nothing booked by the coordinator
    - The failure scenario is when the booking is not removed from the databases and continues to show on checking schedule.

### 4.3 TEST CASE 3

- Writing a blog in the alumni connect feature
- User must be logged in
- Once clicked on the add post feature, corresponding blog entries are filled
- Logged in user makes a POST request to add a blog
- Blog gets displayed in the alumni connect page.

### 4.4 TEST CASE 4

- Registering for Placements
- User Must be logged in
- User clicks on register button
- Test data to be used is corresponding user data from the database.

- Appropriate message to be displayed if he has registered, already registered, or not eligible to register.

### 4.5 TEST CASE 5

- Automated mailing engine
- Coordinator must be logged on
- Coordinator either sends mail to a particular company or uploads the CSV response into the system
- Test Data to be used is the filled out Google Form downloaded as a CSV file
- Two possible scenarios:
    - In case of sending mail, success is when the company receives the mail from the placement office.
    - In case of uploading the CSV, success is when the CSV parsed JSON file is inserted into the company database

### 4.6 TEST CASE 5

- Skill Refinement
- User must be logged on
- User take the test for a particular subject
- Test Data to be used is the scraped interview questions JSON file, which is parsed into the system
- After the completion of the test, a successful scenario is when the user's profile gets updated with the new scores.

## 5 TEST RESULTS

| CASE # | CTIVE/functionality | Kind | STATUS | | N |
|---|---|---|---|---|---|
| 1 | COORDINATOR SCHEDULING AND BOOKING SLOTS | UNIT TEST / INPUT VALIDATION / WHITE BOX TESTING | SUCCESSFUL | NISHANT RAVI SHANKAR | 17/11/2019 |

| | | | | | |
|---|---|---|---|---|---|
| **2** | **COORDINATOR CANCEL BOOKING AND SLOTS** | **NCTIONALITY TEST / WHITE BOX TESTING** | **SUCCESSFUL** | **NISHANT RAVI SHANKAR** | **17/11/2019** |
| **3** | **ALUMNI CONNECT FEATURE** | **FUNCTIONALITY TEST / WHITE BOX TESTING** | **SUCCESSFUL** | **NINAAD R RAO** | **17/11/2019** |
| **4** | **ONE CLICK REGISTRATION** | **SYSTEM WIDE TESTING / FUNCTIONALITY TEST / WHITE BOX TESTING** | **SUCCESSFUL** | **NINAAD R RAO** | **17/11/2019** |
| **5.** | **AUTOMAIL** | **SYSTEM WIDE TESTING / FUNCTIONALITY TEST / WHITE BOX TESTING** | **SUCCESSFUL** | **MIDHUSH MANOHAR T.K.** | **17/11/2019** |
| **6.** | **SKILL CHECKER** | **UNIT TEST / INPUT VALIDATION / WHITE BOX TESTING** | **SUCCESSFUL** | **MIDHUSH MANOHAR T.K.** | **17/11/2019** |

# 6.0 Requirement Traceability Matrix

| Sl. No | Req Id | Brief Desc | Architecture Ref Section | Design Ref | Code File Ref | Unit Test Cases | Function/ System test cases |
|---|---|---|---|---|---|---|---|
| 1 | | Student Information | FR1 | 2.2.3 | students/ views.py | 5.4 | - |
| 2 | | Seating allotment for shortlisted students | FR2 | 2.2.3,2.2.4 | students/ views.py and coordinator/views.py | 5.1 | 5.1 |
| 3 | | Placement Statistics | FR3 | 2.2.3 | students/ views.py | 5.4 | - |
| 4 | | Placement Policies | FR4 | 2.2.3 | templates/homepage.html | 5.4 | - |
| 5 | | Maintaining a calendar system for students | FR5 | 2.2.3 | students/ views.py | 5.1 | 5.1 |
| 6 | | Company information and job search filter | FR6 | 2.2.3 | students/ views.py | - | - |
| 8 | | One click registration | FR8 | 2.2.3 | students/ views.py | 5.4 | 5.4 |
| 9 | | Scheduling Event | FR9 | 2.2.3,2.2.4 | coordinator/views.py | 5,1,5.4 | 5.1,5.4 |
| 10 | | Company Search Engine | FR10 | 2.2..3 | students/ views.py | - | - |
| 11 | | Alumni Connect | FR11 | 2.2.3 | students/ views.py | 5.3 | - |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 12 | | Skill Refinement | FR12 | 2.2.3 | students/ views.py | 5.6 | |
| 13 | | Automail | FR13 | 2.2.4 | coordinator/views.py | 5.5 | 5.5 |