WOW – WORD OF WINGS

Project report submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

Vikrant Singh Rana (141293)

UNDER THE SUPERVISON OF

Mr. Ankur Jat

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CANDIDATE’S DECLARATION

I hereby declare that the work which is being presented in this project work entitled “WORD OF WINGS (WOW)” in partial fulfillment of the requirements for the award of the degree of B.Tech in CSE, JUIT, Waknaghat is an authentic record of my own work carried out during the period January to May 2018 under the supervision and guidance of Mr Ankur Jat

I have not submitted the matter embodied in this project work anywhere for the award of any degree or diploma

Vikrant Singh Rana

ACKNOWLEDGEMENT

It is my proud privilege to express my profound gratitude to the entire management of JUIT and professors of the institute for providing me with the opportunity to avail the excellent facilities and infrastructure. The knowledge and values inculcated have proved to be of immense help at the very start of my career.

I also thank **Prof. Dr. Satya Prakash Ghrera, FBCS, SMIEEE**
Professor, Brig (**Retd.) and Head, Dept. of CSE and IT** , Department of Computer science and Engineering, Jaypee University of Information Technology, for consent to include copyrighted pictures as a part of our report. We thank all the people for their help directly and indirectly to complete our project.

I would like to thank **Grappus Technologies Pvt. Ltd.** for providing me with an opportunity to pursue my industrial training, as it is an important part of the B .Tech course and it is the one that exposes you to the industry standards and makes you adapt yourself to the latest trends and technologies. At the same time, it gives an experience of working on a project.

I express my sincere gratitude to **Mr. Ankur Jat** for his inspiration, constructive suggestion, mastermind analysis and affectionate guidance in my work, without which this project work completion would have been impossible for me. Sincere thanks to all my seniors and colleagues at **Grappus** for their support and assistance throughout the project.

Vikrant Singh Rana

ABSTRACT

The present state of the system is absent. There are no such concepts or programs that the schools take part in separately for the improvement of the reading abilities of the students or for the training of the teaching staff regarding the same. Even if a school, does include a practice in its curriculums, there is no general structure that the other schools could follow.

Maintaining such a program and all its different aspects is a tedious process and WOW takes over this process and simplifies it. Integration of WOW with the school’s curriculum creates a flow that makes sure that the teachers are up to date with the latest learning practices and the students are taking part in such trainings since the beginning of their educational life. It inspires love for reading using literature that is child-friendly and contextual. It makes reading a choice, not a compulsion.

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Introduction About the Company

Grappus Technologies Pvt Ltd. was founded in 2013 by Dhruv Goel and Anuj Birla with the aim of taking the Mobile App Development and the overall user experience of a product to a new level in India. Backed by Ex-IITians and NSIT-ians, Grappus has worked with many big brands such as Mercedes-Benz, PVR Cinemas, Star India (powered by Tata), Uber, Aon Hewitt and more. In addition to this, Grappus has been recently featured in Top 50 Mobile App Companies in India.

Since the beginning, Grappus has aimed at building products that tackle real world problems, but with style. Every project that Grappus has worked on has a certain design essence and user experience that keeps the clients happy and makes them come back for more.

With 46 employees, Grappus has a dedicated team for each project and each team multi-tasks on multiple projects. Apart from project team, there are teams based on the part of the project that they work on such as Front End Team, Backend Team, iOS Team, Android Team, Testing Team and Design Engineering Team. Grappus focuses not only on Designing and Development of products, but it also provides specialised services based on Testing, Estimation, Maintenance and Research.

For each project, the entire team selected is given a proper briefing about the requirements of the clients and the timelines in order. Grappus believes in delivering faster, on time applications with superior quality to the clients, hence justifying the meaning of delivering sprints.

1.2 Problem Statement

Some points highlighting the current state of art:

- There is no structured current system.
- Schools understand the importance of including such a program but lack resources to do so.
- Teachers are not motivated enough to consider this program an extra grooming activity for the students.
- Lack of seriousness towards training of interpersonal skills at school level.

The present state of the system is absent. There are no such concepts or programs that the schools take part in separately for the improvement of the reading abilities of the students or for the training of the teaching staff regarding the same. Even if a school, does include a practice in its curriculums, there is no general structure that the other schools could follow.

Maintaining such a program and all its different aspects is a tedious process and WOW takes over this process and simplifies it. Integration of WOW with the school's curriculum creates a flow that makes sure that the teachers are up to date with the latest learning practices and the students are taking part in such trainings since the beginning of their educational life. It inspires love for reading using literature that is child-friendly and contextual. It makes reading a choice, not a compulsion.

1.3 Proposed Solution

The Indian Education System needs a lot of improvement and it is continually on a path of growth ever since technology started playing a vital role in learning. With this improvement and rapid growth in mind, Stones2Milestones is an organisation that strives to 'Create a Nation of readers'.

Since English is a major international language, the ability to read fluently in English is a life- skill that is crucial to academic success and to participate in the global arena. However, for several reasons, Indian children are trailing behind in their ability to read and comprehend English and we can no longer ignore the research. A child who doesn't read at grade level expectancy by class 4 will never catch up!

The goal is to change this even for children in non-English environments. We want to make learning to read a conscious design of curriculum. The focus is on enabling the skill to make reading easy and also on developing the will to make reading enjoyable. As the window for this intervention is from age 3 to age 9, we integrate this program with the early school curriculum.

This is what led to the birth of '**Wings of Words (WOW)'**. WOW is a reading instruction program that is offered at different grades and different levels. It integrates with the school's curriculum. This project strictly addresses the urgent need of a reading program in our schools. It also focuses on enhancing the learning process from the perspectives of the Student, Teacher and the Parents.

1.4 Deliverables

As a Backend Developer on the project, I was responsible of designing the Database models, development of the APIs and implementation of the flow of the application as required by the client. This involved working on API Deliverables, Setting Up Servers, using 3rd Party Services and APIs and maintaining the database.

S. No.	**Phase**	**Deliverables**	**Page No.**
1.	Requirement Analysis	Use Case	59
2.	System Design	Database Schema	21
3.	System Design	Data Flow Diagrams (Level 0 and Level 1)	61
5.	Construction and Testing	Test Cases	46

6.	Unit Testing	Test Results	47

Table 1.1

Amazon Web Services (AWS) Server: This deliverable was handled using Amazon Web Services (AWS). The project has been decomposed into 5 interdependent micro services, each hosted on an Amazon Web Services (AWS) EC2 Instance. It also involved acquiring elastic IP Addresses for each of these servers and allowing them to interact with each other.

Database Setup: The project uses NoSQL powered MongoDB that allows flexible storage of data in an application due to absence of relationships and faster indexing of large unstructured data. The database is hosted on Amazon Web Services' (AWS) EC2 instance that can only be accessed by any one of the 5 Micro Services in the project.

Firebase: Firebase by Google is a cloud based mobile and web application development platform. It provides large number of services used in the project including Crashalytics and Firebase Cloud Messaging for Push Notifications.

API Deliverables: Backend Development of this project has been implemented using Django for Python, NodeJS with ExpressJS and MongoDB. A total of 121 API Endpoints have been delivered and more are in the development process. The same API Code base is used for a Multi-Functional Admin Panel/CMS and for both iOS and Android Mobile Applications.

Redis Caching Server: Since the amount of data the application will be dealing with is large and the response time needed to be minimised, Redis Caching Server have been set up with the backend to ensure that the API has faster data access and the project works seamlessly.

MSG91 SMS Service: MSG91 is an SMS Messaging service that can be incorporated with the backend APIs and can be used for sending SMS to the users of the project. This service provides special OTP Requesting and Verification operations with IVR Call facility for verifying the authenticity of the users. This service is also used in this project for sending SMS Notifications to registered users.

AWS S3 Service: Amazon Web Services (AWS) provides cloud storage service called S3 that is built to store and retrieve any amount of data from anywhere. The project provides an API Endpoint in which the user can upload multiple files using a single endpoint and in response, would receive an array of URLs generated by S3 after uploading the files on clouds storage.

AWS SES Service: Amazon Web Services (AWS) Simple Email Service (SES) is a cloud email service which is used for sending emails for the users of application/CMS. The SES API is used for sending email notifications to the registered users of the application and is also used for Email Verification of the registered user.

SendX Service: SendX provides growth tools and email marketing automation platform which is integrated with the Backend API to add contacts onto the SendX Service on every user registration. The SendX allows custom automations to be set up so that each user is greeted with a welcome email when it joins WOW and all marketing emails can be send to custom list of users.

CHAPTER 2
PROJECT DESCRIPTION

2.1 System Interface

The Wings of Words system has the following interfaces for the Users interaction:

1. **Login Screen**
   a. This interface allows the users of the system to provide appropriate credentials to the WOW system.
   b. This interface also allows the users to login using the OTP Verification mechanism.
   c. On submission of the data on the screen, the User gets logged into the system or is shown an appropriate error message.
2. **Dashboard Screen**
   a. In this screen, as a *Teacher or Coordinator,* the User can view its progress and reports on different Sections that it teaches throughout the system.
   b. In this screen, as a *WowAdmin,* the User can view the reports of activities associated to the schools to which the WowAdmin is assigned.
   c. In this screen, as a *SuperAdmin,* the User can view the reports on the activities associated with the system.
3. **Learn Screen**
   a. In this Screen, the User will be able to see create the Readings and Trainings with Sessions
   b. The User will be able to see all the Readings and Trainings with Sessions
4. **Notification Screen**
   a. In this screen, the User will be able to create new Notifications.
   b. The Notification to be created will be sent to the selected Users for a selected school.
   c. In this screen, the User will be able to see all the Notifications that they've created and received.
5. **Help Centre Screen**
   a. In this screen, the User can create new queries for another user.
   b. After creating a query, the User can chat with respect to that query.
   c. The user can view all the queries that they've created and the chat messages for each query.
6. **Content Screen**
   a. In this screen, the User can create new Plans, Milestones, Miles, Trainings with Sessions and Readings.
   b. The user can see the content that they have created.
7. **School Screen**
   a. In this screen, the User can add new Schools.
   b. In this screen, the User can see the schools that are a part of the system, assigned to them or the school that they teach in.
   c. The User can select the school and update its details.
   d. The User can select the school and view all classes and sections in that school.
   e. The User can select the school and view all the teachers and coordinators of the School.
   f. The User can create new Coordinators and Teachers of the school.
8. **Interaction Screen**
   a. In this screen, the User can Create an Interaction.
   b. The Interaction will be created for a School and involve other Users that conduct the interaction and other Users that take part in Interactions.
   c. The user can filter the Interactions by Schools and Users involved.

2.2 System Specifications

2.2.1 Hardware Requirements

For development, Each server instance used for hosting the project is an AWS Instance Type T2.Medium and has the following hardware specifications:

- Architecture
    - X86 or X86-64 bit hardware architecture
    - Intel Xeon processor with compatible Motherboards
- Processing Power
    - 3.3-megahertz (MHz) processor or faster
- Memory
    - At least 4 gigabytes (GB) of RAM.
- Secondary Storage
    - At least 30 gigabytes (GB) of available space on the hard disk

For Users, following are the hardware specifications:

- Architecture
    - X86 or X86-64 bit hardware architecture
- Processing Power
    - 1-gigahertz (GHz) processor or faster
- Memory
    - At least 1 gigabytes (GB) of RAM.
- Secondary Storage
    - At least 150 megabytes (MB) of available space on the hard disk

2.2.2 Software Requirements

WOW uses the following software requirements **for development**:

- Python 3.6
- NPM
- Node JS
- Express Js
- Mongo DB

For Users, following are the software requirements:

- Android Marshmallow or Above
- iOS 10 or above.
- Web Browser with an active internet connection.

2.3 Methodologies and Tools used

Node Js (v5.9.3) with Express

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

It also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent. All APIs of Node library are asynchronous, that is, non-blocking. It essentially means a Node based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node helps the server to get a response from the previous API call. It is suitable for writing fast and scalable APIs.

Django (v2.0.0) and REST APIs
Django is a free and open source web application framework written in Python. It is very widely used for writing highly scalable and data oriented Web Applications and APIs according to the Representational State Transfer (REST) architectural pattern. RESTful web services are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work
best on the Web.

Django Rest Framework
Django Rest Framework (DRF) is powerful, sophisticated, and surprisingly easy to use. It offers an attractive, web browse-able version of your API, and the option of returning raw JSON. The Django Rest Framework provides powerful model serialization, display data using standard function based views, or get granular with powerful class based views for more complex functionality.

MongoDB (v10)
MongoDB is a cross-platform document-oriented database system. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favour of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. Released under a combination of the GNU Affero General Public License and the Apache License, MongoDB is free and open source software.

React JavaScript (v16.3.2)
React is an open-source JavaScript library which is used for building user interfaces specifically for single page applications. It's used for handling view layer for web and mobile apps. React also allows us to create reusable UI components.
React allows developers to create large web applications which can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in application. This corresponds to view in the MVC template. It can be used with a combination of other JavaScript libraries or frameworks, such as Angular JS in MVC.

React Native (v0.55)
React Native is a JavaScript framework for rendering mobile application in iOS and Android. React is a Facebook's JavaScript library for building user interfaces which targets mobile platforms. So now developers can make mobile applications using this JavaScript library which can be shared between platforms that makes it easy to develop in both iOS and Android.
By using React Native, you can use the same code for deployment on iOS as well as on Android. This takes you to huge saving in development time and money. As per the calculations 90% of the code can be reused between Android and iOS which saves a lot of time when building a multi-platform applications.

Lifecycle Model
Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methods or Agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals.
Scrum is a subset of Agile. Scrum is most often used to manage complex software and product development, using iterative and incremental practices. Scrum significantly increases productivity and reduces time to benefits relative to classic “waterfall” processes. Scrum processes enable organizations to adjust smoothly to rapidly-changing requirements, and produce a product that meets evolving business goals

2.3.1 Requirement Phase

The Requirements phase focuses on what the system will do in an effort that views all stakeholders, including sponsors and potential users, as important sources of information.

- **Use Case model**: Use case models enable you to explore how users will work with your system. Use cases are been defined of different work Scenarios under this model. A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system.
- **Initial Domain Model:** A domain model identifies fundamental business entity types and the relationships between them. Since the system uses NoSQL Database for its data modelling and data storage practices. A general Database Modelling has been defined which showcases the JSON structures of the entities of the system.

Requirements have defined for first versions which includes a system which can get Users in the system, user level and school level interactions. The requirements have been categorized below under Functional, Usability, Reliability requirements.

Functional Requirements:
- Designing and mapping database models and active entities throughout the system.
- Setting up servers.
- Defining flows and implementing APIs of User login and registration, listing and User Profile.
- Defining flows and implementing APIs of School creation and Listing.
- Defining flows and implementing APIs of Content creation and listing. Content involves Plan, Milestones, Miles, Trainings with Session and Readings.
- Defining flows and implementing APIs of Class and Section creation and Listing.
- Defining flows and implementing logic of User authentication and authorization.
- Defining flows and implementing APIs of Notification creation and listing, delivery of notifications from Push, SMS and Email mechanism.

- Defining flows and implementing APIs for Query creation, listing, message sending, message listing and Push Notifications for each message sent.
- Defining logics and implementing APIs for Assigning a Plan to School, Removing a Plan from a School and Listing all Plans assigned to a School.
- Defining logics and implementing APIs for Assigning a Milestone to a Section and list all Milestones assigned to a Section.
- Defining logics and implementing APIs for Assigning a Teacher to a milestone in a Section and list all Sections of a Teacher.

Non-Functional Requirements:

As well as functional requirements which are define specific behaviour or functions of our system we can identify non-functional requirements that can be used to judge the operation of our system. Generally we can identify non-functional requirement of our system WOW under these categories:

- Smooth User Interface
- Data Security
- Platform Availability
- Continuity
- Optimized Performance

Usability Requirements

These requirements typically specify validating the information entered by the user on web pages. Also, authenticating each user interacting with the system.

Reliability Requirements

Any or all states of different interactions by the users must be saved persistently in the system.

2.3.2 Design Phase

Design phase commences after the requirements finalised and frozen for the first sprint. The design phase attempts to uncover various entities involved in the system and their associated behaviour and also the interfaces that would be provided by the system.

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts.

Design elements describe the desired system features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo-code, and a complete database modelling of the entities involved. These design elements are intended to describe the system in sufficient detail, such that UX, UI and Designing Engineers may design Screens for web and Mobile Devices approved by the client with minimal additional input design. Database Schema and Data Flow Diagrams for the system are developed to understand the in-depth flow and functionality of the system. This phase also involved identifying the actors of the system.

Actors involved in the system:

- **Super Admin**
  - The Super Admin in typical is the owner of the system. He/she has access and permissions to perform all the tasks and can view any content that is a part of the system.
  - The Super Admin can create Content (Plan, Milestone, Miles, Readings and Trainings with Sessions).

- They can also create any other user profile of any type (Super Admin, Wow Admin, Coordinator, Teacher).
- They can create Schools, Classes and Sections. They can assign Wow Admins to schools.
- They can assign any Plan to a School.
- They can also assign any Milestone to a Section and a Teacher for a milestone of that section.
- They can create Notifications for all users of the WOW System.

- **Wow Admin**
  - The Wow Admin is a user that can only be created by a Super Admin and has some specific operation permissions.
  - A Wow Admin can create another User (Wow Admin, Coordinator, Teacher).
  - They cannot create Content but can read it.
  - They cannot create Schools but they can see the list of schools that they have been assigned to.
  - They can create classes and sections for the schools that they've been assigned to.
  - They can see the list of the plans and milestones assigned to a school assigned to them.
  - They can assign a Milestone to a Section of a Class in a School.
  - They can assign a Teacher to a milestone in a Section.

- **Teacher**
  - The Teacher is a School Level User and can only be created by either Super Admin or Wow Admin.
  - They cannot create any other user.
  - They can view the Sections that they are teaching and they can only view the Plans and Milestones that those sections are assigned.
  - They can chat with a Wow Admin assigned to their school by creating a Query.
  - They can see the notifications that have been sent to them.
  - They can see the Trainings with Sessions and Readings that are created by either Super Admin and Wow Admin.
- **Coordinator**
  - The Coordinator is a School Level User and can only be created by either a Super Admin or a Wow Admin.
  - The coordinator cannot create any user.
  - The coordinator have all functionalities of Teacher along with the ability to assign a teacher to a milestone in a section.

2.3.3 Development Phase

The System is developed as 3 tier architecture with bottom up strategy defined as follows:

2.3.3.1 Presentation Layer

This layer contains the provision of logging in and exploring the user's custom dashboard. The user inputs the corresponding information on the html page the details are then transferred in JSON format to frameworks like Nodejs and Django for Python.

Basically it consists of components that provide a common bridge into the core business logic encapsulated in the business layer.

2.3.3.2 Business Logical Layer

The request been made on the web page is passed to the Nodejs or Django for Python and the control goes onto the respective method according to the routes been provided. The methods contain the business logic regarding the login verification, JWT token authentication, school creation, content creation and other activities.

2.3.3.3 Access Layer

The data from the models is then saved into MongoDB using Mongoose for Nodejs or Mongoengine for Django. By using either Mongoose or Mongoengine, the database can be queried using simple Query Dictionaries similar to JSON. The data is then saved in MongoDB which is deployed on an AWS Instance.

2.3.4 Implementation Phase

The System is implemented on 2 type safe framework and the database is maintained under MongoDB. User Interacts via a web Browser, or via Smart Phone apps by the means of a Web service hosted on the server. The server receives the call on exposed API's, interprets it and calls corresponding business methods and replies back after processing the requested operation. To access the data the APIs connect to the MongoDB. The data fetched from the DB is converted into data model using Mongoose or Mongoengine classes, the data of the case class is then stored in the database.

2.3.5 Testing Phase

WOW has been developed with parallel testing processes. In order to increase reliability and to reduce ripple effect on code, testing is not limited to the process of executing a program or application with the intent of finding software bugs (errors or other defects). It is focused on the appropriation of the UI and UX along with the accuracy and stability of Backend APIs. Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- Meets the requirements that guided its design and development
- Works as expected
- Can be implemented with the same characteristics
- Satisfies the needs of Client.

2.4 User Characteristics

- **Super Admin**
  - The Super Admin in typical is the owner of the system. He/she has access and permissions to perform all the tasks and can view any content that is a part of the system.
  - The Super Admin can create Content (Plan, Milestone, Miles, Readings and Trainings with Sessions).
  - They can also create any other user profile of any type (Super Admin, Wow Admin, Coordinator, Teacher).
  - They can create Schools, Classes and Sections. They can assign Wow Admins to schools.
  - They can assign any Plan to a School.

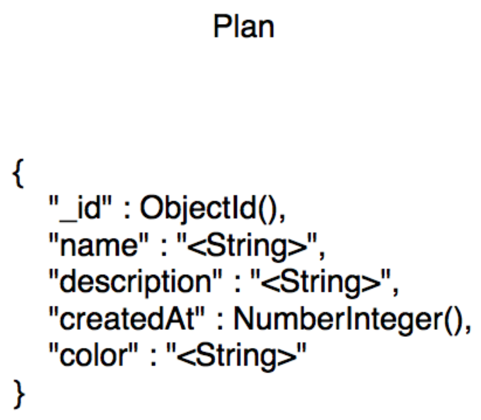
    - They can also assign any Milestone to a Section and a Teacher for a milestone of that section.
    - They can create Notifications for all users of the WOW System.
- **Wow Admin**
    - The Wow Admin is a user that can only be created by a Super Admin and has some specific operation permissions.
    - A Wow Admin can create another User (Wow Admin, Coordinator, Teacher).
    - They cannot create Content but can read it.
    - They cannot create Schools but they can see the list of schools that they have been assigned to.
    - They can create classes and sections for the schools that they've been assigned to.
    - They can see the list of the plans and milestones assigned to a school assigned to them.
    - They can assign a Milestone to a Section of a Class in a School.
    - They can assign a Teacher to a milestone in a Section.
- **Teacher**
    - The Teacher is a School Level User and can only be created by either Super Admin or Wow Admin.
    - They cannot create any other user.
    - They can view the Sections that they are teaching and they can only view the Plans and Milestones that those sections are assigned.
    - They can chat with a Wow Admin assigned to their school by creating a Query.
    - They can see the notifications that have been sent to them.
    - They can see the Trainings with Sessions and Readings that are created by either Super Admin and Wow Admin.
- **Coordinator**
    - The Coordinator is a School Level User and can only be created by either a Super Admin or a Wow Admin.
    - The coordinator cannot create any user.
    - The coordinator have all functionalities of Teacher along with the ability to assign a teacher to a milestone in a section.

CHAPTER 3
SYSTEM DEVELOPMENT

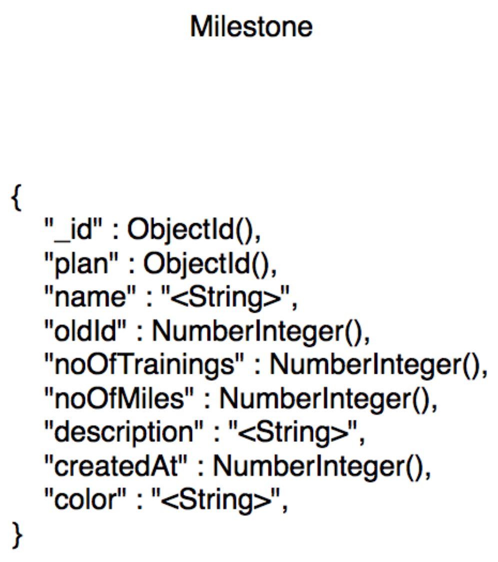
3.1 Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the data related to each other is referenced. It formulates all the constraints that are to be applied on the data. A database schema can be divided broadly into two categories:
Physical Database Schema − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
Logical Database Schema − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.
Since WOW uses MongoDB as its database, MongoDB stores data in a binary representation called BSON (Binary JSON). MongoDB documents tend to have all data for a given record in a single document, whereas in a relational database information for a given record is usually spread across many tables.

3.1.1 Database Structure in JSON


Plan

```
{
   "_id" : ObjectId(),
   "name" : "<String>",
   "description" : "<String>",
   "createdAt" : NumberInteger(),
   "color" : "<String>"
}
```


Milestone

```
{
   "_id" : ObjectId(),
   "plan" : ObjectId(),
   "name" : "<String>",
   "oldId" : NumberInteger(),
   "noOfTrainings" : NumberInteger(),
   "noOfMiles" : NumberInteger(),
   "description" : "<String>",
   "createdAt" : NumberInteger(),
   "color" : "<String>",
}
```

Mile

```
{
    "_id" : ObjectId(),
    "milestone" : ObjectId(),
    "plan" : ObjectId(),
    "oldId" : NumberInteger(),
    "tags" : ["<String>"],
    "assessment" : [Object],
    "training" : [ObjectId()],
    "section" : [Object],
    "planName" : "<String>",
    "milestoneName" : "<String>",
    "name" : "<String>",
    "description" : "<String>",
    "createdAt" : NumberInteger(),
}
```

Reading

```
{
    "_id" : ObjectId(),
    "name" : "<String>",
    "oldId" : NumberInteger(),
    "tags" : ["<String>"],
    "section" : [Object],
    "titleImage" : "<String>",
    "coverImage" : "<String>",
    "authorImage" : "<String>",
    "authorName" : "<String>",
    "description" : "<String>",
    "likes" : NumberInteger(),
    "visitedCount" : NumberInteger(),
    "createdAt" : NumberInteger(),
    "__v" : NumberInteger(),
    "estimatedTimeToRead" : Object
}
```

Training

```
{
   "_id" : ObjectId(),
   "name" : "<String>",
   "tags" : ["<String>"],
   "duration" : {
      "seconds" : NumberInteger(),
      "minutes" : NumberInteger(),
      "hours" : NumberInteger()
   },
   "noOfVideos" : NumberInteger(),
   "peopleInTraining" : NumberInteger(),
   "description" : "<String>",
   "createdAt" : NumberInteger(),
   "titleImage" : "<String>",
   "coverImage" : "<String>",
   "color" : "<String>",
   "__v" : NumberInteger()
}
```

Sessions

```
{
   "_id" : ObjectId(),
   "name" : "<String>",
   "training" : ObjectId(),
   "section" : [Object],
   "description" : "<String>",
   "createdAt" : NumberInteger(),
   "__v" : NumberInteger()
}
```

User

```
{
   "_id" : ObjectId(),
   "name" : "<String>,
   "phone" : NumberLong(),
   "password" : "<String>",
   "userType" : "<String>",
   "email" : "<String>",
   "isEmailVerified" : <Boolean>,
   "isContactNumberVerified" : <Boolean>,
   "username" : "<String>",
   "createdAt" : NumberLong(),
   "teachesTo" : [],
   "activeDevices" : [<String>],
   "address" : {
      "city" : "<String>",
      "pincode": NumberInteger(),
      "locality": "<String>",
      "country": "<String>",
      "state":"<String>"
   },
   "last_login" : NumberLong()
}
```

School

```
{
    "_id" : ObjectId(),
    "name" : "<String>",
    "email" : "<String>",
    "phone" : NumberLong(),
    "academic" : {
        "start_date" : "<String>",
        "end_date" : "<String>"
    },
    "address" : {
        "city" : "<String>"
    },
    "affiliationBoard" : "<String>",
    "createdAt" : NumberLong(),
    "displayPicture" : "<String>",
    "foundDay" : "<String>",
    "milestones" : [ObjectId()],
    "planId" : [ObjectId()],
    "uniqueKey" : NumberLong(),
    "summer_vacations" : {
        "start_date" : "<String>",
        "end_date" : "<String>"
    },
    "winter_vacations" : {
        "start_date" : "<String>",
        "end_date" : "<String>"
    },
    "final_exams" : {
        "start_date" : "<String>",
        "end_date" : "<String>"
    },
    "schoolType" : ObjectId(),
    "curriculum_books" : "<String>",
    "remarks" : "<String>"
}
```

Class

```
{
   "_id" : ObjectId(),
   "name" : "<String>",
   "school" : ObjectId()
}
```

ClassSection

```
{
  "_id" : ObjectId(),
  "name" : "<String>",
  "noOfStudents" : NumberLong(),
  "class_" : ObjectId(),
  "teach" : [Object]
}
```

School
Type

```
{
  "_id" : ObjectId(),
  "type" : "<String>"
}
```

Teacher Training Progress

```
{
  "_id" : ObjectId(),
  "sessionId" : ObjectId(),
  "teacherId" : ObjectId(),
  "trainingId" : ObjectId(),
  "createdAt" : NumberInteger(),
  "status" : "<String>"
}
```

Teacher Feedback

```
{
   "_id" : ObjectId(),
   "teacherId" : ObjectId(),
   "type" : "<String>",
   "planId" : ObjectId(),
   "milestoneId" : ObjectId(),
   "mileId" : ObjectId(),
   "review" : NumberInteger(),
   "tags" : [<String>],
   "createdAt" : NumberLong()
}
```

Query

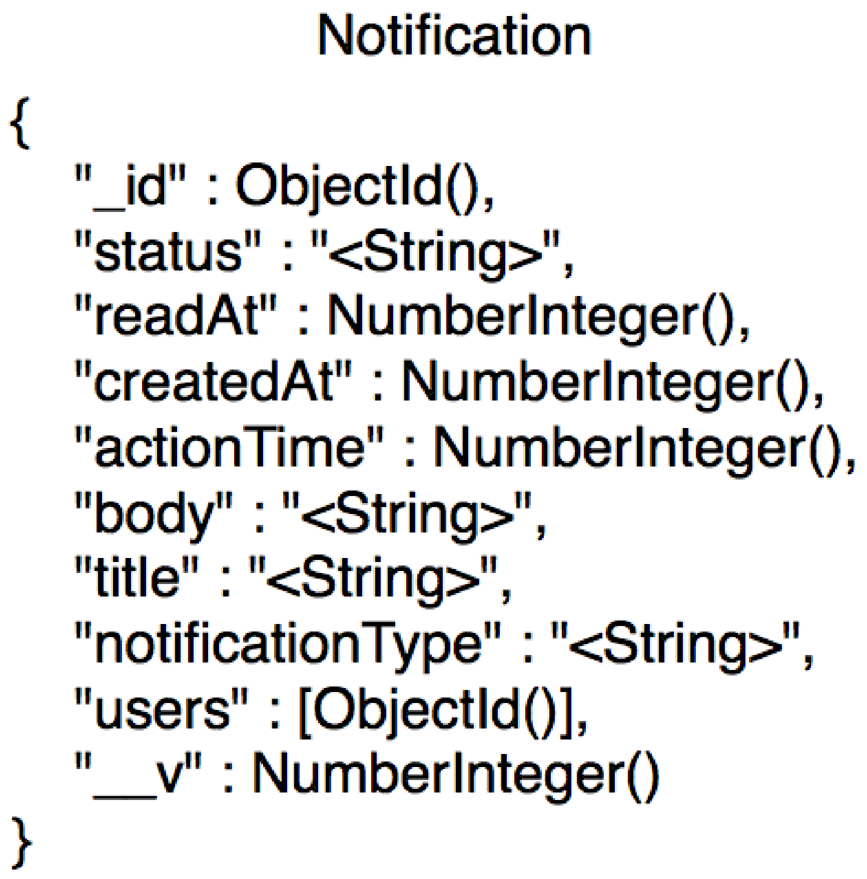
```
{
   "_id" : ObjectId(),
   "createdBy" : ObjectId(),
   "createdFor" : ObjectId(),
   "status" : "<String>",
   "lastMessage" : "<String>",
   "query" : "<String>",
   "createdAt" : NumberInteger(),
   "__v" : NumberInteger(),
   "updatedAt" : NumberInteger()
}
```

Message

```
{
    "_id" : ObjectId(),
    "media" :,
    "type" : "<String>",
    "createdAt" :NumberInteger(),
    "sentBy" : ObjectId(),
    "message" : "<String>",
    "queryId" : ObjectId(),
    "__v" : NumberInteger()
}
```

Device

```
{
    "_id" : ObjectId(),
    "createdAt" :NumberInteger(),
    "deviceType" : "<String>",
    "deviceToken" : "<String>",
    "deviceId" : "<String>",
    "userId" : ObjectId(),
    "__v" : NumberInteger()
}
```



Notification

```
{
    "_id" : ObjectId(),
    "status" : "<String>",
    "readAt" : NumberInteger(),
    "createdAt" : NumberInteger(),
    "actionTime" : NumberInteger(),
    "body" : "<String>",
    "title" : "<String>",
    "notificationType" : "<String>",
    "users" : [ObjectId()],
    "__v" : NumberInteger()
}
```

Figure 3.1

3.2 Input Output Design

3.2.1 Login

Purpose - User accesses login page or the login screen on the app. If the user already has an account he/she provides his/her phone and password to login which further directs the user to his dashboard if the details provided are correct and the user has verified its account.

Description of fields

- **User phone** – It is the unique phone of the user.
- **User password** – It is the password for authenticating the account.

Validation

- **User email Id –** Must be 10 digit long and start from 7, 8 or 9 and must exist for a user.
- **User password –** Must be between 6-14 characters. It must match the password selected at sign up.

The dashboard of a user has following option:

- Content
- Interactions
- Notifications
- Progress
- Help Centre

3.2.2 Dashboard

Purpose - User reaches dashboard after successful login. Here user can access the content that they are allowed to see, Interactions that take place for his school, view the notifications that have been sent to them. If the user has been assigned some Sections from its school, they can view their progress in their sections. The user can also see the Queries that they have raised and the chat thread associated with those queries, or even raise new queries.

3.2.3 School

Purpose – In this view, the Super Admin or a Wow Admin can view the profile of the school and assign it new Plans.

Description of fields

- Drop down List of Plans.

3.2.4 Classes of a School

Purpose – User can see the list of classes in a School along with Milestone assigned to a Class, Teacher assigned to that Milestone, Total Miles Done, Total Number of Students. From the same view, a New Section for a Class can be added. To create a new Section, Milestone needs to be selected, a class must be selected, name must be selected and number of students must be provided.

Description of fields

- **Milestone** – It is a drop down list containing milestones of the active Plan.
- **Class** – It is the class for which the Section will be made.
- **Section Name** – It is the name of the section.

- **Number of Students** – It is the number of Students in that section.
- **Teacher (Optional)** – It is a drop down of the teacher from the School that may be assigned to that section in the school.

Validation

- **Number of Students –** Must be a number.
- **Section Name –** Must be a String.

3.2.5 Content

Purpose – Users can view the content of Miles of Milestones of a Selected Plan.

3.3 Use Case Description

3.3.1 Login

Use Case 1	Login
Actors(s)	Super Admin, Wow Admin, Teacher, Coordinator
Preconditions	1) The system must be up and functional. 2) The user must have a registered account on Wings Of Words 3) User must be verified.
Type	Primary
Description	The user logins into the WOW System.
Sequence	Actor Action User click on login button on WOW System home page *OR* User opens the WOW Android or iOS App for the first time on their phone: 1) User enter valid username 2) User enter valid password 3) User clicks on Login button System Response 1) User logins into the WOW System
Alternative Sequence	1) Error message is displayed if credentials are wrong 2) Error message is displayed if the User is not verified.
Post Conditions	User Dashboard Page Loaded into browser *OR* User dashboard screen displayed on the Android or iOS App.

Table 3.1

3.3.2 OTP Verification

Use Case 2	OTP Verification
Actors(s)	Super Admin, Wow Admin, Teacher, Coordinator

Preconditions	1) The system must be up and functional. 2) The user must have a registered account on Wings Of Words.
Type	Primary
Description	The user uses OTP Verification mechanism to Verify its account and login into the system.
Sequence	<u>Actor Action</u> User click on Login via OTP on WOW System home page *OR* User taps the Login via OTP button on Login Screen in the Android or iOS App for the first time on their phone: 1) User enter valid phone 2) User enter valid OTP received on the phone 3) User clicks on verify button <u>System Response</u> 1) User logins into the WOW System
Alternative Sequence	1) Error message is displayed if phone is wrong. 2) Error message is displayed if OTP is wrong.
Post Conditions	User Dashboard Page Loaded into browser *OR* User dashboard screen displayed on the Android or iOS App.

<u>Table 3.2</u>

3.3.3 Registration

Use Case 3	Registration
Actors(s)	Super Admin, Wow Admin, Teacher, Coordinator
Preconditions	1) The system must be up and functional. 2) An existing user of the system should be logged in. 3) The creation must only be done by another existing and logged in User of Wings of Words.
Type	Primary
Description	A logged in User creates an account for another User.
Sequence	Actor Action User click on Add New Profile on a School's Teachers Page *OR* User click on Add New Profile on WOW Admins Page: 1) User enter valid data 2) User clicks on create button. System Response 1) New User is created in the WOW System.
Alternative Sequence	1) Error message is displayed if user details fail validation. 2) Error message is displayed if the User Phone already registered.
Post Conditions	Current page is refreshed to display the updated list of Users.

Table 3.3

3.3.4 Create Plans

Use Case 4	Create Plans
Actors(s)	Super Admin, Wow Admin
Preconditions	1) The system must be up and functional. 2) An existing user of the system should be logged in. 3) The creation must only be done by either Super Admin or Wow Admin
Type	Primary
Description	A logged in User creates a new Plan.
Sequence	<u>Actor Action</u> User click on Add New Plan on Content Screen’s Mile Tab 1) User enter valid data 2) User clicks on create button. <u>System Response</u> 2) New Plan is created in the WOW System.
Alternative Sequence	Error message is displayed if plan details fail validation.
Post Conditions	Current page is refreshed to display updated list of Plans.

Table 3.4

3.3.5 Create Milestones

Use Case 5	Create Milestones
Actors(s)	Super Admin, Wow Admin
Preconditions	1) The system must be up and functional. 2) An existing user of the system should be logged in. 3) The creation must only be done by either Super Admin or Wow Admin 4) Plan chosen should already exist.
Type	Primary
Description	A logged in User creates a new Milestone in a Plan.
Sequence	<u>Actor Action</u> User click on Add New Milestone on Content Screen's Mile Tab 1) User choses a Plan from list of Plans available 2) User enter valid data 3) User clicks on create button. <u>System Response</u> 1) New Milestone is created for the chosen Plan in the WOW System.
Alternative Sequence	1) Error message is displayed if milestone details fail validation. 2) Error message is displayed if the Plan chosen doesn't exist.
Post Conditions	Current page is refreshed to update the list of Milestones for the chosen Plan.

Table 3.5

3.3.6 View List of Plans

Use Case 6	View list of Plans
Actors(s)	Super Admin, Wow Admin, Teacher, Coordinator
Preconditions	1) The system must be up and functional. 2) An existing user must be logged in.
Type	Primary
Description	A logged in User views the list of all Plans.
Sequence	<u>Actor Action</u> User click on Content Tab on sidebar <u>System Response</u> A list of Plans filtered according to the logged in User.
Alternative Sequence	NA
Post Conditions	Current page is loaded with a List of Plans filtered according to the user.

Table 3.6

3.3.7 View List of Milestones

Use Case 7	View list of Milestones
Actors(s)	Super Admin, Wow Admin, Teacher, Coordinator
Preconditions	1) The system must be up and functional. 2) An existing user must be logged in. 3) An existing Plan must be chosen.
Type	Primary
Description	A logged in User views the list of all Milestones in a chosen Plan.
Sequence	<u>Actor Action</u> User click on a Plan in the Content Tab <u>System Response</u> A list of Milestones of the chosen Plan filtered according to the logged in User.
Alternative Sequence	Error message is displayed if the Plan chosen is incorrect.
Post Conditions	Current page is loaded with a List of Milestones of a Plan filtered according to the user.

Table 3.7

3.3.8 Create Miles

Use Case 8	Create Miles
Actors(s)	Super Admin, Wow Admin
Preconditions	1) The system must be up and functional. 2) An existing user of the system should be logged in. 3) The creation must only be done by either Super Admin or Wow Admin 4) Plan chosen should already exist. 5) Milestone chosen should already exist.
Type	Primary
Description	A logged in User creates a new Mile for a Milestone in a Plan.
Sequence	<u>Actor Action</u> User click on Add New Mile on Content Screen's Mile Tab 1) User choses a Plan from the list of existing Plans 2) User choses a Milestone from the list of Milestones in the chosen Plan 3) User enter valid data 4) User clicks on create button. <u>System Response</u> New Mile is created in the chosen Milestone for the chosen Plan in the WOW System.
Alternative Sequence	1) Error message is displayed if mile details fail validation. 2) Error message is displayed if the Plan chosen doesn't exist. 3) Error message is displayed if the Milestone chosen doesn't exist.
Post Conditions	Current page is refreshed to update the list of Miles in the chosen Milestone for the chosen Plan.

Table 3.8

3.3.9 View a List of Miles

Use Case 9	View list of Miles
Actors(s)	Super Admin, Wow Admin, Teacher, Wow Admin
Preconditions	1) The system must be up and functional. 2) An existing user must be logged in. 3) An existing Plan must be chosen. 4) An existing Milestone must be chosen from the chosen Plan.
Type	Primary
Description	A logged in User views the list of all Miles for a chosen Milestones in a chosen Plan.
Sequence	Actor Action 1) User click on Plan in the Content Tab 2) User clicks on a Milestone from the list of Milestones in the chosen Plan System Response A list of Miles for a chosen Milestone of the chosen Plan filtered according to the logged in User.
Alternative Sequence	1) Error message is displayed if the Plan chosen is incorrect. 2) Error message is displayed if the Milestone of the chosen Plan is incorrect.
Post Conditions	Current page is loaded with a List of Miles for a chosen Milestone of the chosen Plan filtered according to the user.

Table 3.9

3.3.10 Assign a Plan to a School

Use Case 10	Assign a Plan to a School
Actors(s)	Super Admin, Wow Admin
Preconditions	1) The system must be up and functional. 2) An existing user must be logged in. 3) Assigning of a Plan to a School can only be done by a Super Admin or Wow Admin 4) Plan to be assigned must already exist. 5) School to whom the plan is to be assigned must already exist.
Type	Primary
Description	A logged in User assigns an existing Plan to an existing School.
Sequence	<u>Actor Action</u> 1) User selects a School 2) User selects a Plan 3) User confirms assigning the plan to that school. <u>System Response</u> A Plan is assigned to the School.
Alternative Sequence	1) Error message is displayed if the selected Plan is incorrect 2) Error message is displayed if the selected School is incorrect 3) Error message is displayed if the Plan is already assigned to the School
Post Conditions	Current page is reloaded with a updated Plans for the selected School.

Table 3.10

3.3.11 Assign a Milestone of a Plan to a Section

Use Case 11	Assign a Milestone of a Plan to a Section
Actors(s)	Super Admin, Wow Admin, Coordinator
Preconditions	1) The system must be up and functional. 2) An existing user must be logged in. 3) Assigning of a Milestone of a Plan to a Section can only be done by a Super Admin, Wow Admin or Coordinator 4) Milestone to be assigned must already exist. 5) Section to whom the milestone is to be assigned must already exist.
Type	Primary
Description	A logged in User assigns an existing Milestone of a chosen Plan to a chosen Section of a chosen Class in a chosen School.
Sequence	<u>Actor Action</u> 1) User selects a School 2) User selects a Class from the chosen School 3) User selects a Section from the chosen Class 4) User selects a Plan from the chosen School 5) User selects a Milestone from the chosen Plan 6) User confirms assigning the chosen Milestone to the chosen Section. <u>System Response</u> A Milestone is assigned to the Section.
Alternative Sequence	1) Error message is displayed if the selected Plan is incorrect 2) Error message is displayed if the selected Milestone is incorrect 3) Error message is displayed if the selected School is incorrect 4) Error message is displayed if the selected Class is incorrect 5) Error message is displayed if the selected Section is incorrect 6) Error message is displayed if another Milestone from same plan is already assigned to the Section
Post Conditions	Current page is reloaded with a updated Milestone for the selected Section of the chosen Class in the chosen School.

Table 3.11

3.3.12 Assigning a Teacher to a Milestone of a Plan to a Section

Use Case 12	Assign a Teacher to a Milestone of a Plan to a Section
Actors(s)	Super Admin, Wow Admin, Coordinator
Preconditions	1) The system must be up and functional. 2) An existing user must be logged in. 3) Assigning a teacher to a Milestone in a Section can only be done by a Super Admin, Wow Admin or Coordinator 4) Teacher to be assigned must already exist. 5) Section to whom the plan is to be assigned must already exist. 6) Section must have the milestone assigned.
Type	Primary
Description	A logged in User assigns an existing Teacher to a Milestone of a chosen Plan to a chosen Section.
Sequence	<u>Actor Action</u> 1) User selects a School 2) User selects a Class from the chosen School 3) User selects a Section from the chosen Class 4) User selects a Plan from the chosen School 5) User selects a Milestone from the chosen Plan 6) User selects a Teacher from the list of teachers of that school 7) User confirms assigning the chosen Teacher to the milestone in the chosen Section. <u>System Response</u> A teacher is assigned to the milestone in the Section.
Alternative Sequence	1) Error message is displayed if the selected Plan is incorrect 2) Error message is displayed if the selected Milestone is incorrect 3) Error message is displayed if the selected School is incorrect 4) Error message is displayed if the selected Class is incorrect 5) Error message is displayed if the selected Section is incorrect
Post Conditions	Current page is reloaded with a updated Section.

Table 3.12

3.3.13 View progress and Upcoming miles for a teacher with respect to a Plan

Use Case 13	View progress and Upcoming miles for a teacher with respect to a Plan
Actors(s)	Super Admin, Wow Admin, Teacher, Coordinator
Preconditions	1) The system must be up and functional. 2) An existing user of the system should be logged in. 3) The Teacher must already exist. 4) The Plan must already exist
Type	Primary
Description	A logged in User views an existing Teacher's progress with respect to an existing Plan.
Sequence	<u>Actor Action</u> User click on Teacher's Profile on a School's Teachers Page *OR* User can login on the Android and iOS App: 1) User selects a School. 2) User selects a Plan. 3) User selects a Teacher <u>System Response</u> A detailed description of Upcoming Miles, Completed Miles, Total Miles and Progress per section are returned.
Alternative Sequence	NA
Post Conditions	NA

Table 3.13

3.3.14 View Training Progress, Upcoming Sessions and Reading resources for the Teacher

Use Case 14	View Training Progress, Upcoming Sessions and Reading resources for the Teacher
Actors(s)	Super Admin, Wow Admin, Teacher, Coordinator
Preconditions	1) The system must be up and functional. 2) An existing user of the system should be logged in.
Type	Primary
Description	A logged in User (Teacher) views existing Training Progress, Upcoming Sessions and Reading resources.
Sequence	Actor Action User can login on the Android and iOS App and tap on Learn Screen. System Response A detailed description of Training Progress, Upcoming Sessions and Reading resources are returned.
Alternative Sequence	NA
Post Conditions	NA

Table 3.14

CHAPTER 4
PERFORMANCE ANALYSIS

4.1 Test Activities

Testability is simply, how easily computer program can be tested. Thoroughly tested code not only checks weather implementation is correct it also reflects the goodwill of the company. WOW had a target to achieve the test coverage of 95 percent or above which was successfully achieved. Test procedures and test case design solve various difficult tasks, some of the traits of a good test case are

- A good test case has a high probability of finding out errors.
- It should be non-redundant.
- It has to be best of breed.
- A good test case should neither be too simple nor too complex.

4.2 Unit Testing

This was the first phase of testing. In unit testing the program were tested separately to ensure their correctness independently. Each unit developed was tested independently without other system components and any interactions with other programs. During this each and every line of code was tested also each conditional statement and loop statements were tested for all its branches to cover maximum branch and statement coverage.

4.2.1 Methodology Used

Unit testing is carried out by the developer in the developer environment only. Manual testing was done. The developers review their code to check whether their respective units under tests behave as expected.

4.2.2 Tools Used

Manual testing was carried out in Django and Nodejs by writing Test Cases for each file of Models, Views and Controllers with unittest library from Python and Chai in Node respectively.

4.2.3 Test Cases

Test Id	Test case Name	Test Case Description	Test Case Input	Test Results		Test case Status
				Expected	Actual	
1	To Verify the functionality of Login button with the valid phone.	User should be able to login into the Wings Of Words system.	Valid input credentials.	User should be able to login.	User logins into Wings of Words system.	Pass
2	Error on Invalid phone.	While login into Wings Of Words system if no user found with entered phone then he should not able to login into Wings of Words system.	Invalid user phone.	Login should fail.	Login Fails.	Pass

Test Id	Test case Name	Test Case Description	Test Case Input	Test Results		Test case Status
				Expected	Actual	
3	To Verify that the functionality of creating a user with non-existing phone is working correctly.	User should be able to create a user with a Unique phone not existing in the Wings of Words system.	User data with unique phone.	User is created.	User is created.	Pass
4	To Verify that the functionality of creating a user with existing phone should give an error.	User should not be able to create another user with existing phone.	User details with existing phone.	User should not be created.	User is not created.	Pass

Test Id	Test case Name	Test Case Description	Test Case Input	Test Results		Test case Status
				Expected	Actual	
5	To test that notifications created are delivered to the listed users	On creating a notification, the users in the list must receive a push notification on all their devices.	Users list, Notification message and title.	Notification should be saved in the database and delivered to all listed users.	Notification is saved in the database and received by all listed users.	Pass
6	To test that Sections are required while creating a Mile for a Milestone in a Plan.	While creating a Mile, sections are required without which a mile cannot be created.	Mile details excluding the sections.	The mile details must contain sections.	The mile details must contain sections.	Pass

Table 4.1

4.3 Integration Testing

4.3.1 Methodology used

All the modules were integrated together and then were checked for errors and whether they work properly or not after integration. After integration it was also checked that the code segments were not repeated multiple times.

4.3.2 Tools used

Not Applicable

4.3.3 Test Cases

Test Id	Test case Name	Test Case Description	Test Case Input	Test Results		Test case Status
				Expected	Actual	
1	Assigning a Plan to a School	To assign a plan to a school, the user has to first login and then select a school to assign a plan to. Then they have to select the plan to be assigned.	Correct Login details were provided and correct school and plan was selected	Plan should be assigned to the school.	New plan is added to the school.	Pass

Test Id	Test case Name	Test Case Description	Test Case Input	Test Results		Test Case Status
				Expected	Actual	
2	Assigning a Milestone of a Plan to a Section in a School not containing that Plan.	To assign a Milestone of a Plan to a Section in a School, the User must first log in and select a Section of a Class in a School and then select a Milestone of a Plan.	Correct login details were provided and Milestone was selected from Incorrect Plan	Milestone should not be assigned to the section.	Milestone is not assigned to the section.	Pass
3	Update Teacher on a Milestone in a Section	To Update the Teacher on a Milestone in a Section, the User must log in and select a Section of a Class in a School and then select a Milestone.	Correct login details were provided, a milestone from a section was selected and teacher to be updated was selected.	New teacher should be assigned to the milestone in the section.	New teacher is assigned to the milestone in the section.	Pass

Test Id	Test case Name	Test Case Description	Test Case Input	Test Results		Test Case Status
				Expected	Actual	
4	To verify that the Teacher doesn’t get any progress from a Plan that he is not assigned.	To verify that the Teacher doesn’t get any progress from a Plan that he is not assigned, the User must first log in and then select an incorrect Plan.	Correct login details were provided and incorrect Plan was used.	No progress should be available.	No progress is available.	Pass
5	To verify that deleting a plan will also remove it from all assigned schools.	To Delete a Plan, the User must log in and select Plan to be Deleted.	Correct login details were provided and a Plan was selected to be removed.	The Plan should be removed from Content page and should not visible in any School.	The Plan is removed from Content page and is not visible in any School.	Pass

Table 4.2

4.4 System Testing

4.4.1 Functional Testing

4.4.1.1 Methodology used

Under this the whole system is tested by the development team. Basically, all functionalities as per requirements are tested here. Functional Testing usually describes what the system does.

4.4.1.2 Tools used

Not Applicable. Manual testing was performed.

4.4.1.3 Test Cases

Test Id	Test case Name	Test Case Description
1.	Super Admin has complete control	Super Admin should have complete control over Wings of Words System
2.	Wow Admin can create new content and assign it to Schools	Wow Admin can create new content and assign it to Schools
3.	Coordinator should only view content assigned to its School.	Coordinator can only view content assigned to its School.
4.	Teacher should only view content assigned to his Sections.	Teacher can only view content assigned to his Sections.
5.	Wow Admin should send and view notifications only to schools that he is assigned to.	Wow Admin can send and view notifications only to schools that he is assigned to.
6.	Teachers can only view the Notifications they have received.	Teachers can only view the notifications that they have received.
7.	Teachers should not be able to send notifications.	Teachers cannot send notifications.
8.	Teacher should not create any other User.	Teacher cannot create any other User.

Table 4.3

4.5 Implementation

The tests were first implemented on individual units under Unit Testing. Each of Functionalities inputs were divided as Valid or Invalid Inputs and then a combination of these was tested upon the concerned Functional Modules. The results of these inputs are described in test case results area.

Under System Testing the complete Functionality of the system as per the requirements was tested. Each and every requirement was studied and its implementation with the concerned functionality was provided.

CHAPTER 5
CONCLUSION AND DIFFERENCES

5.1 Conclusion

From a proper analysis of the positive points and constraints on the component, it can be safely concluded that the product is a Multi-Platform Cloud Based Application. The application has been visually designed to be easy for its user to interact with. So it is in the best interest of the organization to use such project which their user can use easily. All the tasks can be done by any user of the system without much hassle and any rigid training.
Concluding points are as follows:

- For all the interactive activities that take place within the application, the user is Notified with a Rich Push Notification that is saved in the databases for the user to view later and keep a track record of its interactions.
- For all the Miles that the User marks as done, they can provide a feedback on whether the content of the Mile was good or it could need improvements. The user can also provide a feedback when they complete a session within a Training.
- Search has been incorporated throughout the project to make it easier for the Users to filter out the content that they're seeing. The search filtering can be done in numerous formats which makes it versatile and provides flexibility from the user's perspective to view the data however they like.
- The application has been built and designed so that in future, it can be improved seamlessly without changing much of the user experience and also incorporating different types of users from different countries and background.

5.2 Limitations of the System

Following are some limitations of the Wings Of Words (WOW) system:

- Performance issues in Android and Mobile Applications.
- Currently the application is only in English language.
- Absence of a Demo mechanism to guide new users on the usage of the system.
- Chat Performance within the Help Centre can be improved.

5.3 Future Scope of the System

The live application is on version *v1*. However, there are many new features and functionalities in the pipeline divided and spread throughout the development timeline in the application. Some of the new features and upcoming changes are as follows:

- **Experience sharing**. The users of the system can share their experience in the form of a Text post or a post containing Photos/videos and text. The other users on the platform would be able to view them, like them and comment on them.
- **Assessments (Quizzes or Surveys)**. The users of the system would be able to fill out surveys and quizzes created by the other users of the system, collectively known as Assessments. The creator of the assessment will be able to view the results (in case of quizzes) and total percentage of users selecting an answer (in case of surveys). The quizzes can also be linked to a content available on the system.
- **Multi Language Support**. The application currently is only available in English. The future versions would also allow the users to change their view

of the website and the app to Indian Languages like (Hindi, Marathi, Telugu, Bengali etc.) for the non-English speaking users.

- **Voice and Video call facilities**. The current chat system would be extended to incorporate Video and Voice calls between the users on all platforms.
- **Performance Improvement**. The newer version of the Android and iOS application will have significantly improved performance and would be smaller in size.

5.4 References and Bibliography

Websites:

- Google (http://www.google.com/)
- Stones2Milestones (http://www.stones2milestones.com/)
- Wow Connect (https://www.wowconnect.in)
- Wikipedia (https://wikipedia.org)
- Draw.io (https://draw.io)
- Firebase by Google (https://firebase.google.com)
- Firebase Documentation (https://firebase.google.com/docs/)
- Amazon EC2 Instance Types (https://aws.amazon.com/ec2/instance-types/)
- Django Project (https://www.djangoproject.com)
- NodeJS (https://nodejs.org/en/)
- ExpressJS (https://expressjs.com)

Books:

- **MongoDB: The Definitive Guide** by Kristina Chodorow, O'Reilly Publications, 3rd Edition
- **RESTful Web API Design with Node.JS** by Valentin Bojinov, Packt Publications, 2nd Edition
- The Django Book (https://djangobook.com)
- **AWS Basics: Beginners Guide** by Gordon Wong
- **Redis in Action** by Josiah L. Carlson, Manning Publications,
- **The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform** by Laurence Moroney, 1st Edition

CHAPTER 6
ANNEXURES

6.1 Use Case Diagram

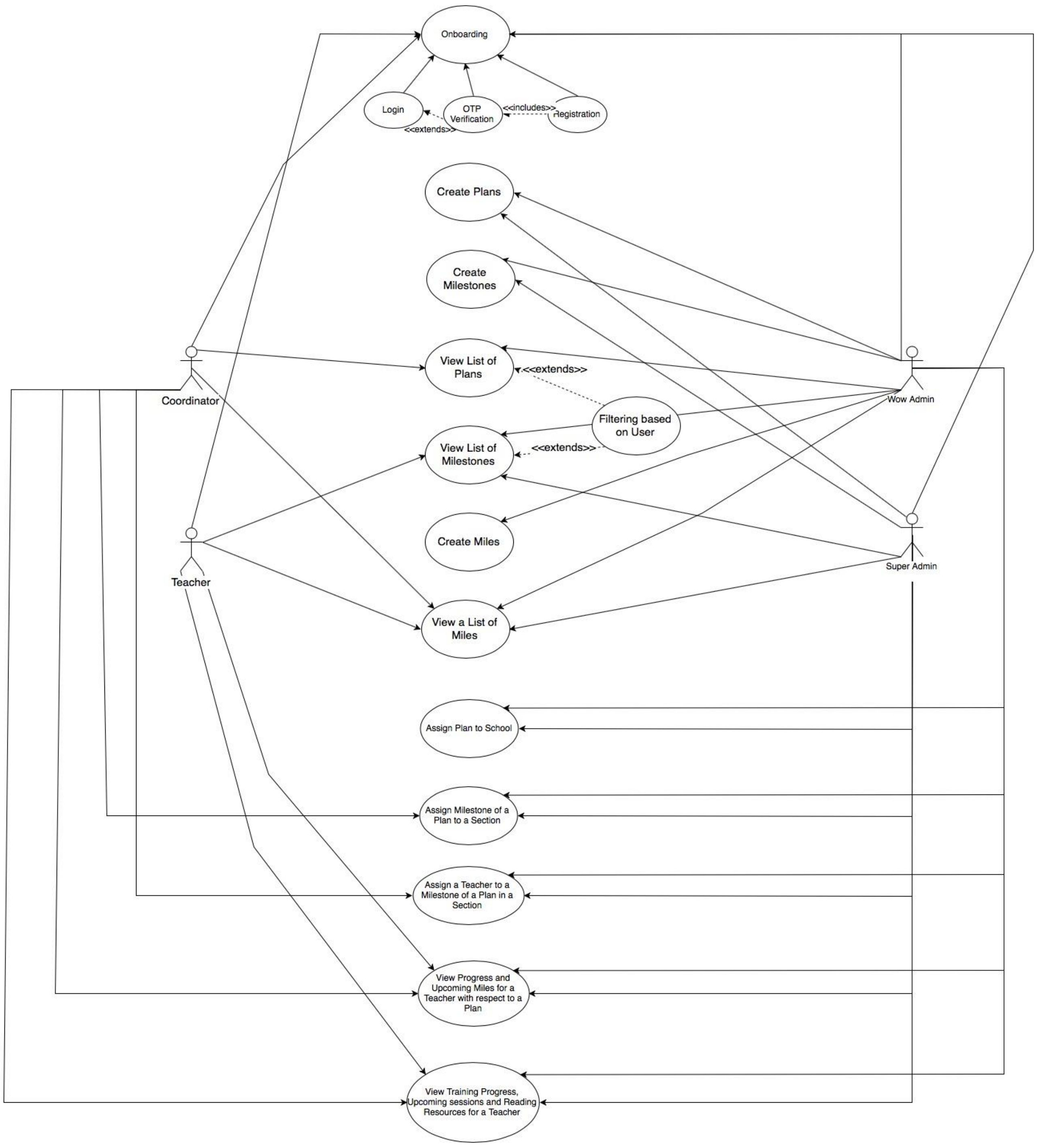


Figure 6.1

6.2 DFD Diagrams

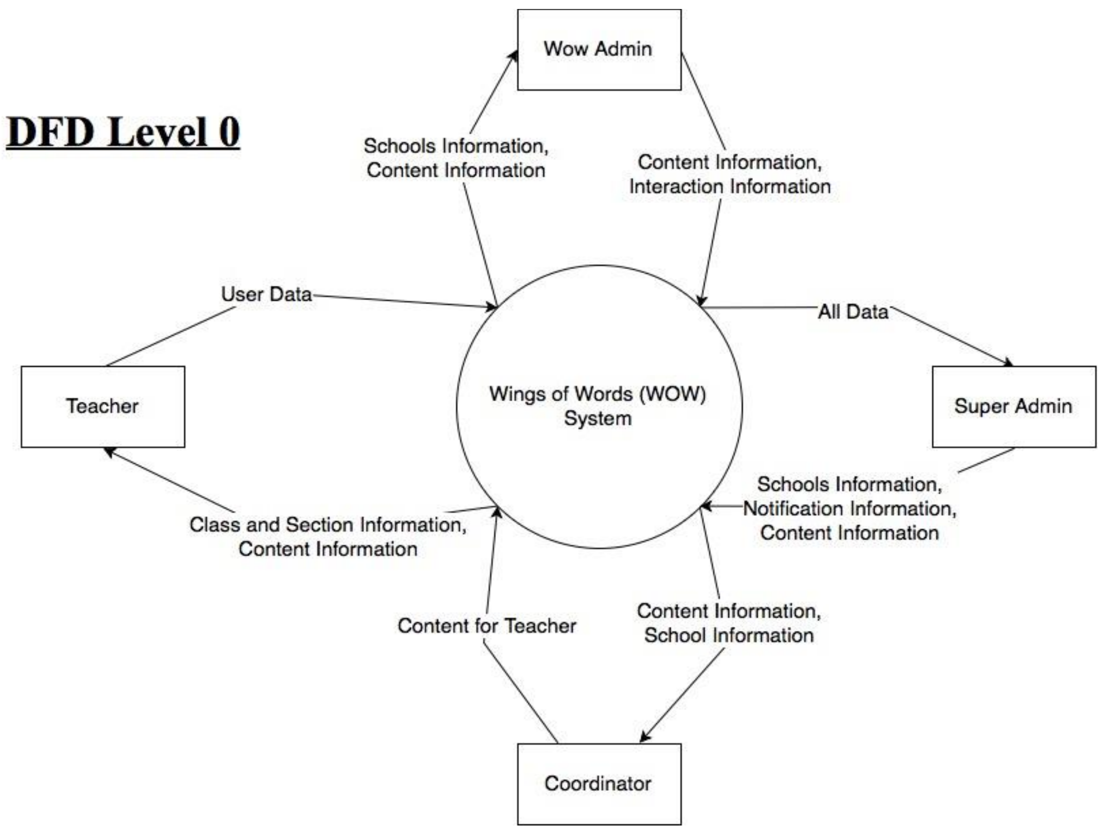

DFD Level 0
Wow Admin
Schools Information, Content Information
Content Information, Interaction Information
User Data
All Data
Teacher
Wings of Words (WOW) System
Super Admin
Schools Information, Notification Information, Content Information
Class and Section Information, Content Information
Content for Teacher
Content Information, School Information
Coordinator

Figure 6.2

DFD Level 1

Abbreviations:

- **Users** - Teacher, Coordinator, Wow Admin, Super Admin
- **Users 1** - Teacher and Coordinator
- **Users 2** - Wow Admin, Super Admin

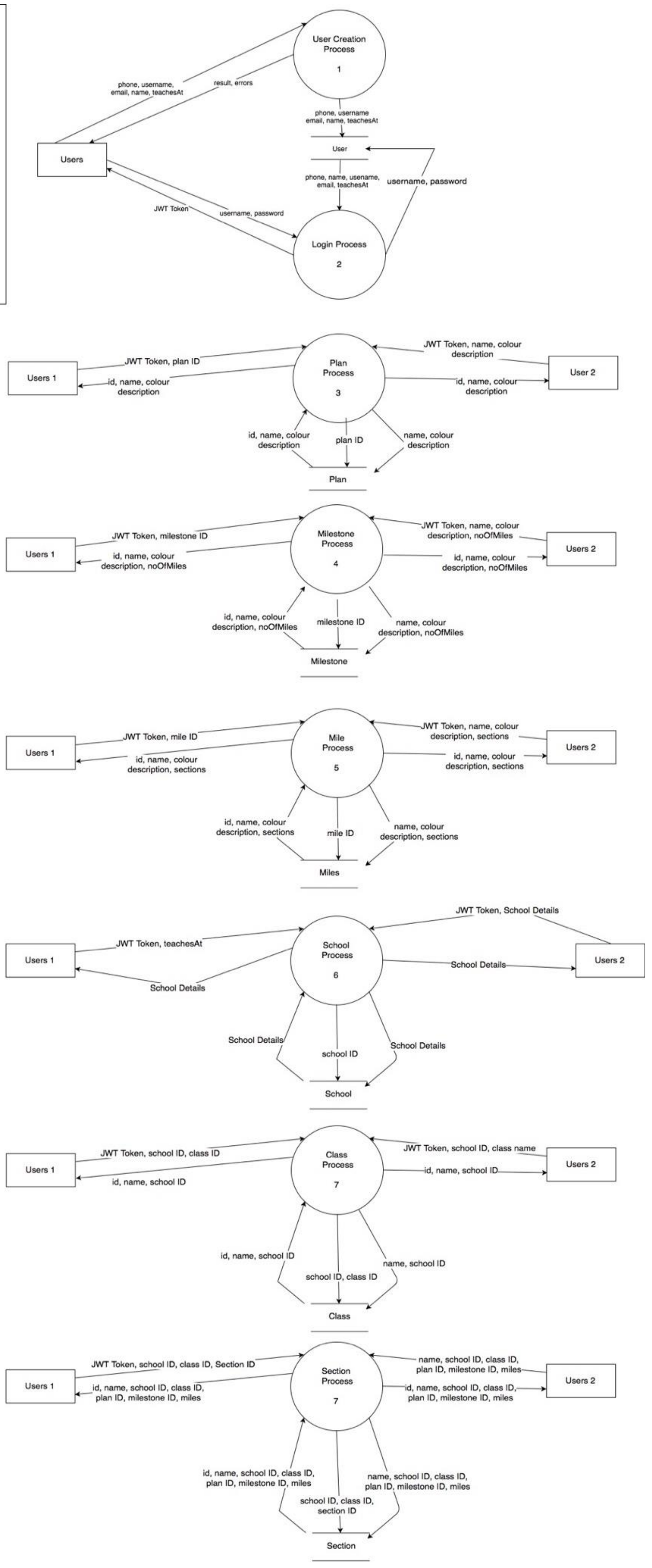
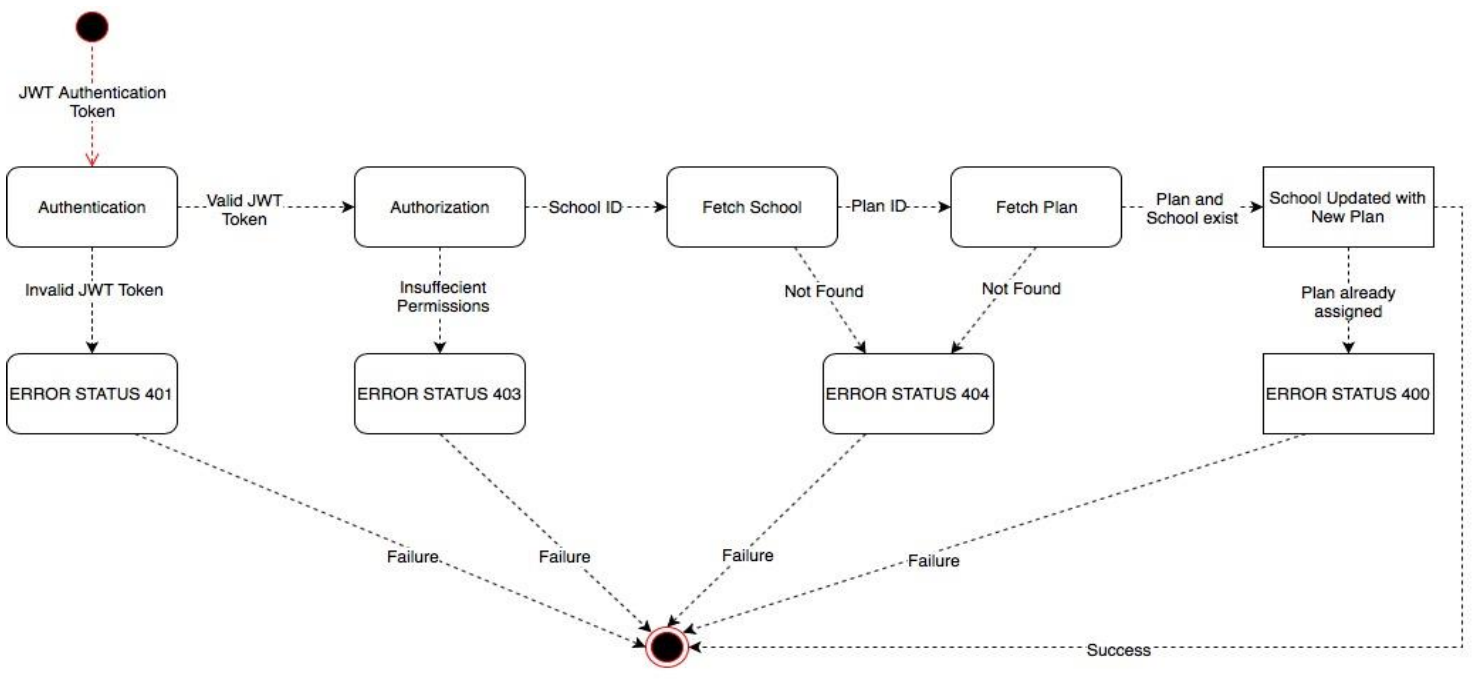

User Creation Process 1
phone, username, email, name, teachesAt
result, errors
phone, username email, name, teachesAt
User
phone, name, usename, email, teachesAt
username, password
Users
JWT Token
username, password
Login Process 2
JWT Token, plan ID
Users 1
id, name, colour description
Plan Process 3
JWT Token, name, colour description
User 2
id, name, colour description
id, name, colour description
plan ID
name, colour description
Plan
JWT Token, milestone ID
Users 1
id, name, colour description, noOfMiles
Milestone Process 4
JWT Token, name, colour description, noOfMiles
Users 2
id, name, colour description, noOfMiles
id, name, colour description, noOfMiles
milestone ID
name, colour description, noOfMiles
Milestone
JWT Token, mile ID
Users 1
id, name, colour description, sections
Mile Process 5
JWT Token, name, colour description, sections
Users 2
id, name, colour description, sections
id, name, colour description, sections
mile ID
name, colour description, sections
Miles
JWT Token, teachesAt
Users 1
School Details
School Process 6
JWT Token, School Details
School Details
Users 2
School Details
school ID
School Details
School
JWT Token, school ID, class ID
Users 1
id, name, school ID
Class Process 7
JWT Token, school ID, class name
Users 2
id, name, school ID
id, name, school ID
school ID, class ID
name, school ID
Class
JWT Token, school ID, class ID, Section ID
Users 1
id, name, school ID, class ID, plan ID, milestone ID, miles
Section Process 7
name, school ID, class ID, plan ID, milestone ID, miles
Users 2
id, name, school ID, class ID, plan ID, milestone ID, miles
id, name, school ID, class ID, plan ID, milestone ID, miles
school ID, class ID, section ID
name, school ID, class ID, plan ID, milestone ID, miles
Section

Figure 6.3

6.3 Activity Diagram



State Transition Diagram for Assigning a Plan to a School

Figure 6.4

6.4 Screenshots

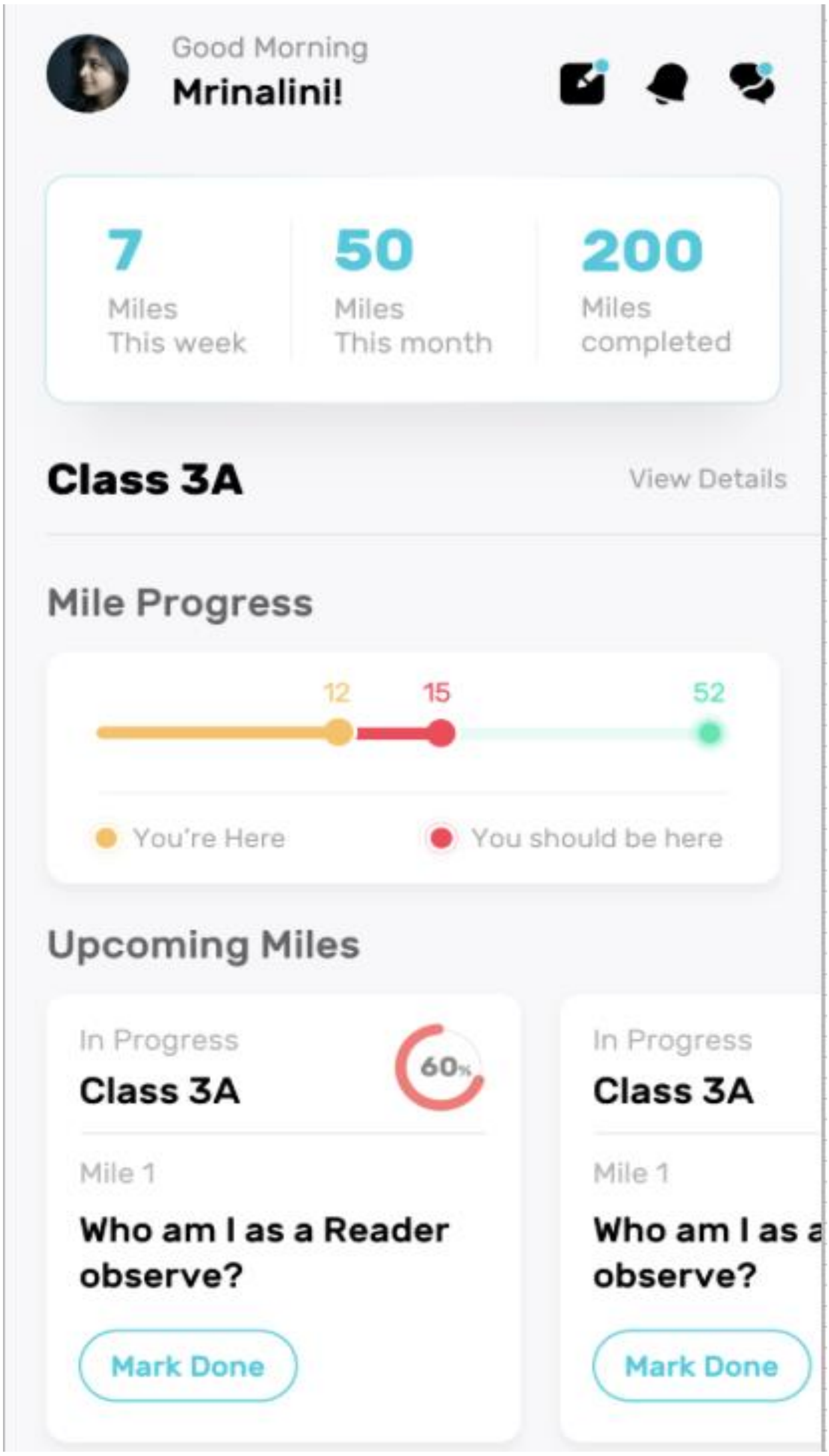
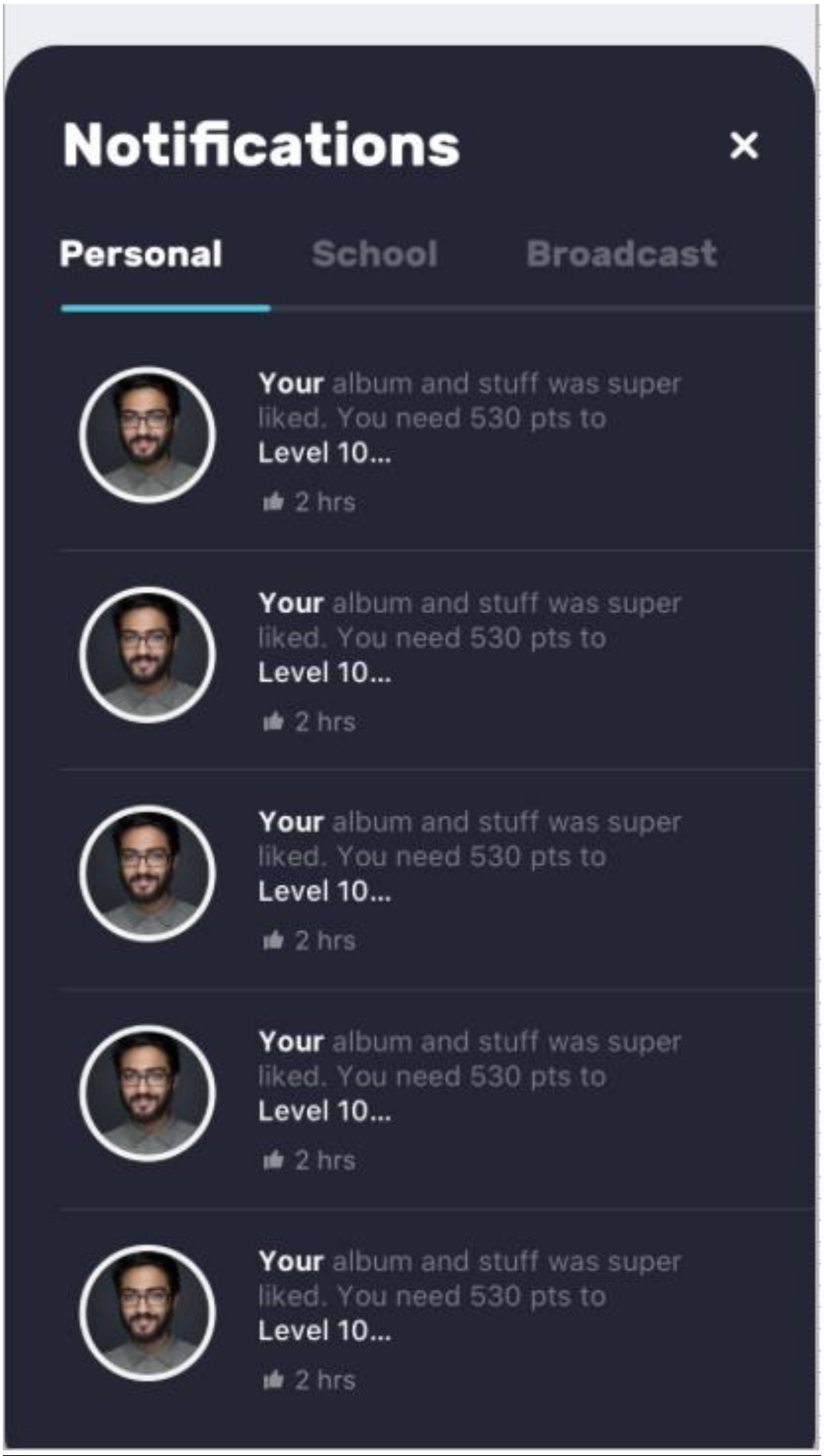


Figure 6.5



Your album and stuff was super liked. You need 530 pts to Level 10...
2 hrs

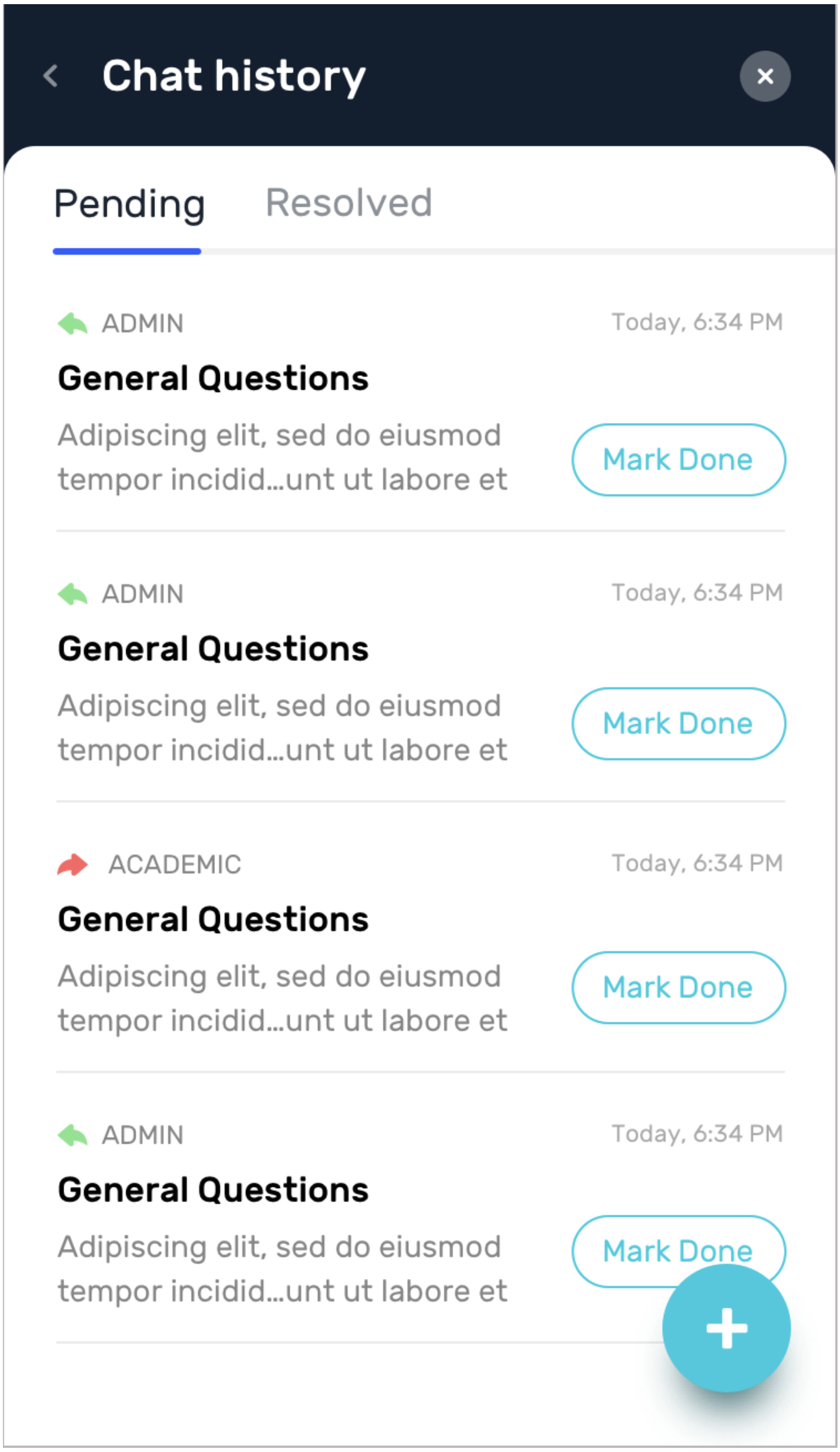
Your album and stuff was super liked. You need 530 pts to Level 10...
2 hrs

Your album and stuff was super liked. You need 530 pts to Level 10...
2 hrs

Your album and stuff was super liked. You need 530 pts to Level 10...
2 hrs

Your album and stuff was super liked. You need 530 pts to Level 10...
2 hrs

Figure 6.6


Chat history

Pending Resolved

ADMIN Today, 6:34 PM
General Questions
Adipiscing elit, sed do eiusmod tempor incidid...unt ut labore et
Mark Done

ADMIN Today, 6:34 PM
General Questions
Adipiscing elit, sed do eiusmod tempor incidid...unt ut labore et
Mark Done

ACADEMIC Today, 6:34 PM
General Questions
Adipiscing elit, sed do eiusmod tempor incidid...unt ut labore et
Mark Done

ADMIN Today, 6:34 PM
General Questions
Adipiscing elit, sed do eiusmod tempor incidid...unt ut labore et
Mark Done

Figure 6.7

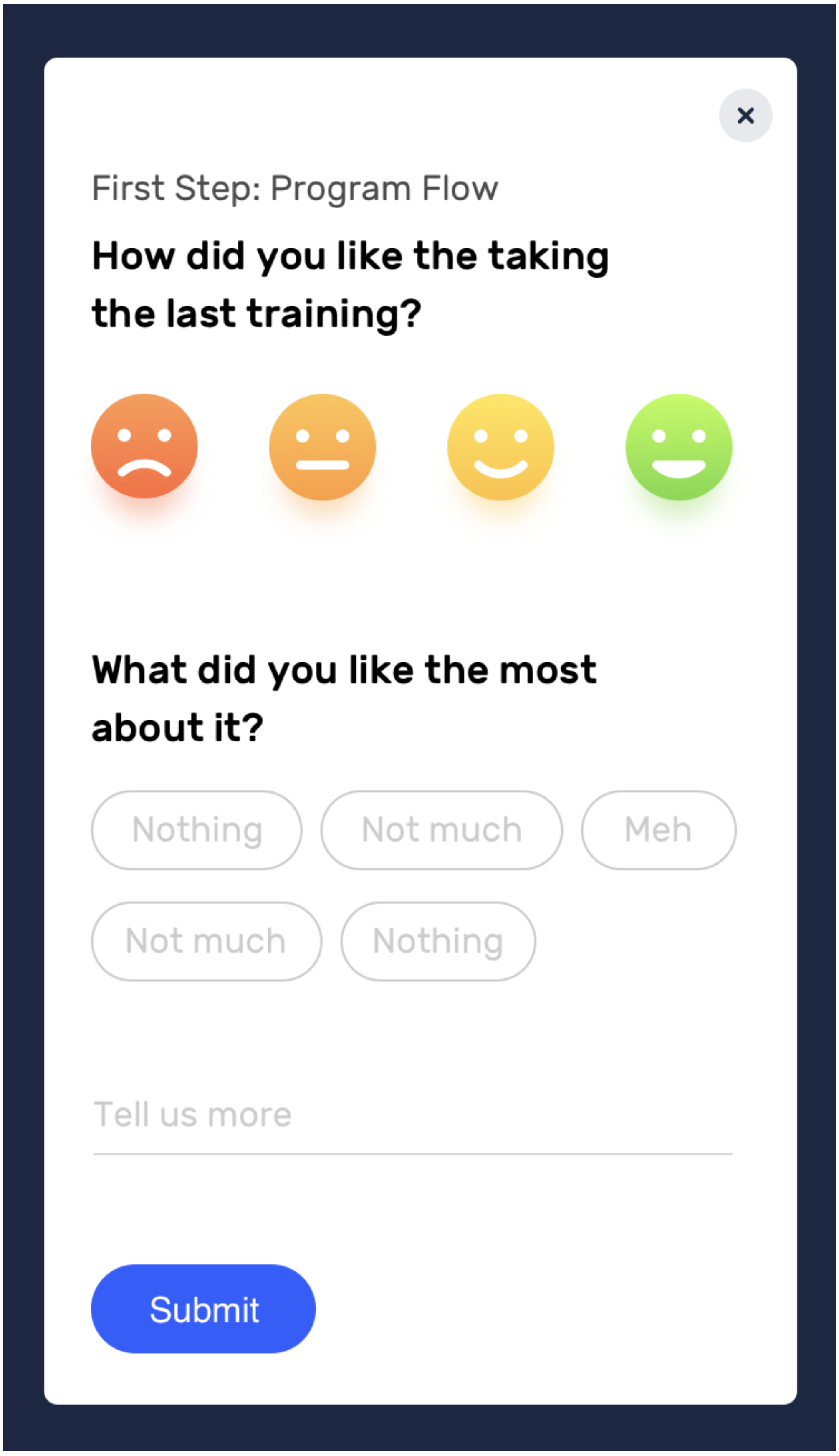
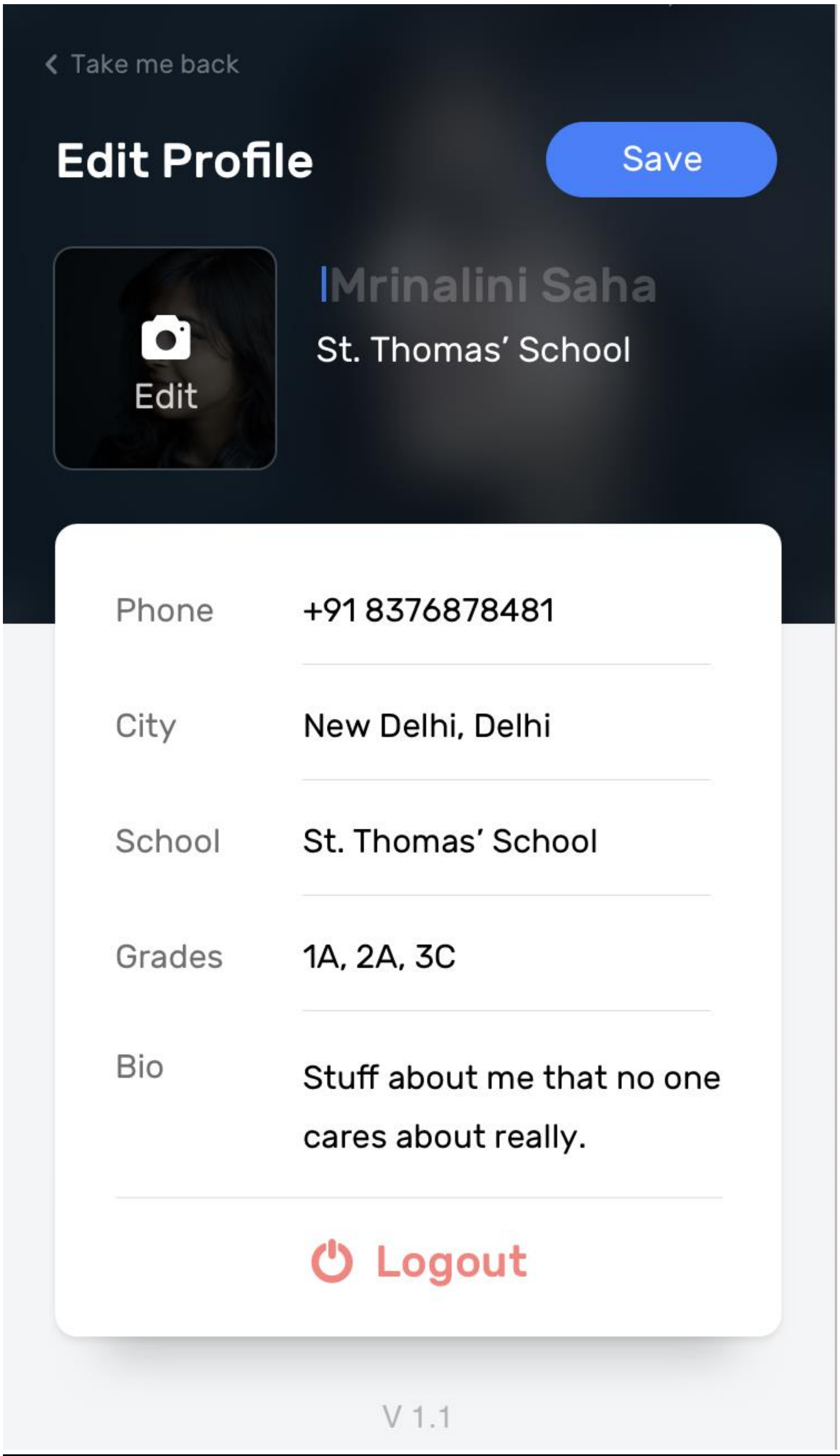

×
First Step: Program Flow
How did you like the taking the last training?
What did you like the most about it?
Nothing
Not much
Meh
Not much
Nothing
Tell us more
Submit

Figure 6.8


Take me back

Edit Profile

Save

Mrinalini Saha

St. Thomas' School

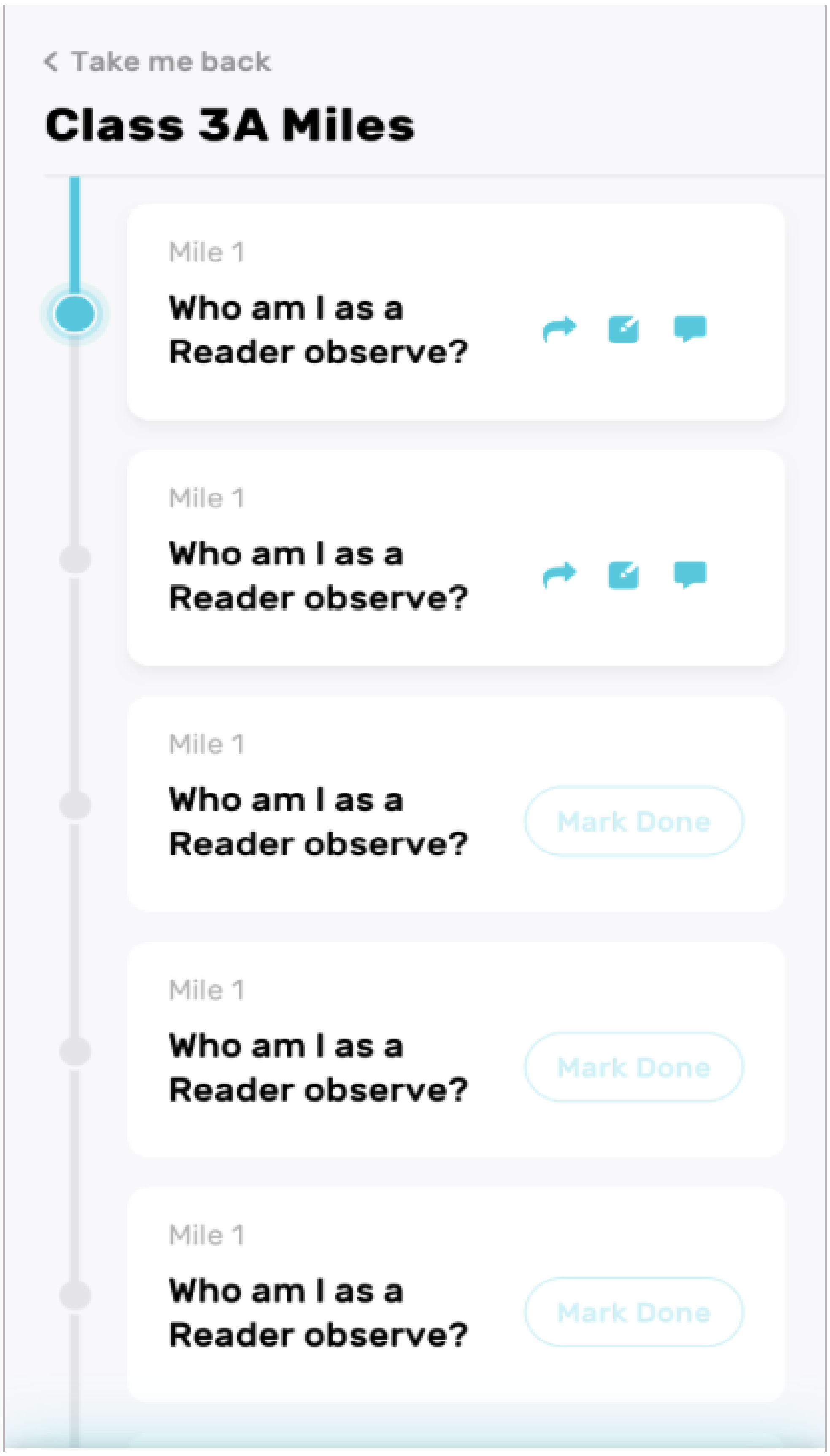
Edit

Phone	+91 8376878481
City	New Delhi, Delhi
School	St. Thomas' School
Grades	1A, 2A, 3C
Bio	Stuff about me that no one cares about really.

Logout

V 1.1

Figure 6.9



< Take me back
Class 3A Miles
Mile 1
Who am I as a Reader observe?
Mile 1
Who am I as a Reader observe?
Mile 1
Who am I as a Reader observe?
Mark Done
Mile 1
Who am I as a Reader observe?
Mark Done
Mile 1
Who am I as a Reader observe?
Mark Done

Figure 6.10