THE UNIVERSITY of York

MSc Thesis

Mobile application development to enhance higher education lectures

Konstantinos Semertzidis

Submitted in partial fulfilment for the degree of

MSc in Computing

Supervisor: Dr. George Despotou

York, 09 September 2013

Number of Words: 25,233 as counted by the MS Word application, excluding the references and the appendices

Abstract

Nowadays, educational technology, even in the early stages, enables curriculum adjustment to meet students' learning and life situations. It also provides alternatives to traditional educational methods and enhances higher education lectures. For example, new technology eliminates spatial and temporal constraints, as students and teachers needn't be in the same classroom, or even within the same area, to exchange information or educational material, but can be anywhere without hampering their work. Furthermore, the operating cost of schools and universities is reduced, since all course materials can be presented using cheap technology, through the screens of electronic devices.

In this project, the goal of our study is three-fold: first, to understand the benefits of learning through mobile devices (m-learning), second, to provide an analysis of principles, patterns of mobile interface design and tactics for solving common mobile development problems, and third, to produce a new m-learning application, based on the findings of this study, which offers direct communication between students and teachers. In order to establish this communication, the application offers poll forms that can host questions with various answer choices. The use of polls also enables the fast collection of students' responses and viewing results on the screens of mobile devices.

Acknowledgements

I would like to thank my supervisor, Dr. George Despotou, for his guidance and support throughout the whole project.

Also, I would like to give my gratitude to my parents for their understanding, support, and faith in me throughout this year. I would not have the opportunity to do this master course without their support.

Statement of ethics

No sensitive data was collected. Users’ emails from the mobile application were not published to anyone. None of these email addresses were used for spamming, phishing or advertising purposes. The aim of the mobile application development was to offer a mobile learning application to the education community.

Table of Contents

List of Figures

List of Tables

List of Listings

Chapter 1: Introduction

The learning process changes over time, much like the lifestyle and the terms and qualifications of employment do. The learning process welcomes more reinforcing components, which contribute to the development of information, communication and interactive environment. For example, the rapid development of web technology, such as the internet, brings e-learning to the fore [1]. E-learning is defined as any use of web technology to create educational activities and experiences. The potential is exciting and amazing.

The emergence of wireless and mobile devices helped e-learning extend to m-learning [2]. One of the positive features of m-learning is that there is a high availability of mobile devices worldwide and this trend continues to grow. It should be taken into consideration that a mobile phone is still a phone so it can transmit audio, text and even video between users, using an m-learning application. Thus, an opportunity is given to develop more interactive applications that will help the promotion of m-learning.

An application that offers operations to support distance learning, such as enabling a student to share his opinion through polls, a teacher to post announcements, share his files with his students and quickly collect the students' responses from polls, can be the start of an m-learning promotion and, thus, is one of our goals in this project. In order to produce a useful m-learning application, we also study the benefits of m-learning and make an analysis of principles and patterns of mobile interface design.

1.1 Aim

The aim of this project is to study the benefits of m-learning, to provide an analysis of principles and patterns of mobile interface design, to provide tactics that solve common mobile development problems and to develop a mobile application for the Android platform that will provide functions which support distance learning and offer direct communication between students and their teachers through the internet.

The application will allow any user to create a course by giving a name for it, and by optionally providing an institution and department name. The owner of a course can add other users to his course, by adding their emails in the invitation form. Also, he can share files with the course members, add announcements and create polls for them.

The members of a course can thus be informed, vote in the given polls and view the polls' results. These functions help teachers collect the students' replies in polls fast, without interrupting the lecture, while students are also free to view the polls' results. Additionally, the function of sharing files with the course members allows students to collect their teacher's files from anywhere, using only the application. Course announcements can be used by the course's owner to inform the members of any news. To provide all the above functions there is the need to create a web server which will host a database for storing the users' data. Also, the communication between the application's users through the web server is done by an application programming interface (API). To better understand how this will work, Figure 1.1 shows the component view of the system.

Furthermore, it is ideally designed to make students more active and motivated in lectures and, last but not least, it can be the start to incite other developers to design more applications that support distance learning through mobile devices.

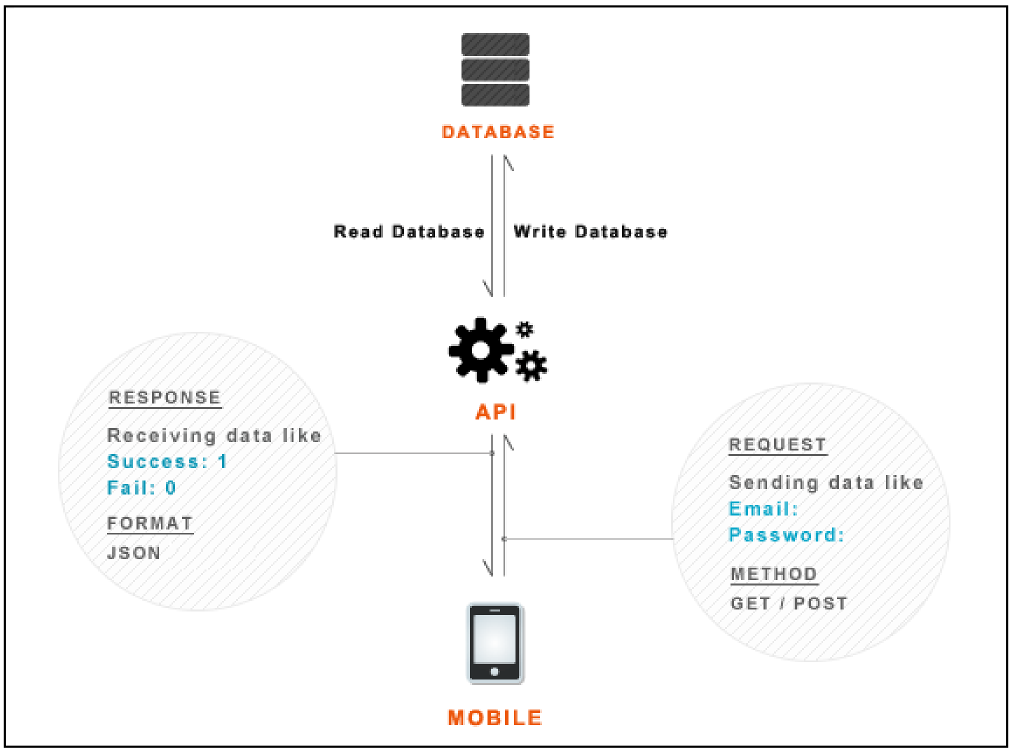


Figure 1.1: Component View of the System

1.2 Project Objectives

In order to achieve these aims, several objectives have to be achieved first. These objectives are:

- Review of literature about the theory of mobile learning.
- Study of design features, principles and design patterns that a mobile application should follow.
- Study of the platform which the mobile application will run on.
- Investigation of tools that will be used to design and develop a mobile application in the Android operating system.
- Provide tactics for solving common problems in Android application development to help new developers.
- Design and development of an Android application that helps teachers establish a direct communication between students and themselves, regardless of their geographical location, without interrupting the lecture.
- Evaluation of the usefulness of the application. The evaluation will be based on students and teachers that will use the mobile application in real time.

1.3 Software Development Process

The software development process is a set of steps that a software program goes through when developed. First in the software development process, the requirements phase outlines the goals of what the program will be capable of doing. Next, the design phase covers how the program is going to be created. The implementation phase is where the programmers and other designers start working on the program. After the developers have a working copy, the testing and verification step can begin to help verify that the program has no errors. During the testing phase, problems found are fixed, until the program meets the company's quality controls. [3]

A software process model consists of an abstract representation of a software process through a specific viewpoint. [4] In this section, some widely used processes are discussed followed by the approach that is adapted in this project.

1.3.1 Waterfall Model

The Waterfall Model was the first software development process to be introduced. It consists of five phases, requirements, design, implementation, testing and finally the deployment and maintenance of the system (Figure 1.2). The waterfall model is a linear-sequential life cycle model, which means that each phase must be completed fully before the next phase can begin. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. It can be easy to understand, use and manage. Its phases are processed and completed one at a time. It can work well for smaller projects where requirements are very well understood. However, it has a high amount of risk and uncertainty. It is a poor model for long and ongoing projects and it is not suitable for projects where requirements are at a moderate to high risk of changing. Also, it cannot produce working software until late into the life cycle. Lastly, when the application is in the testing phase, it is very difficult to go back and change something that was not well-thought out in the concept phase. [5]

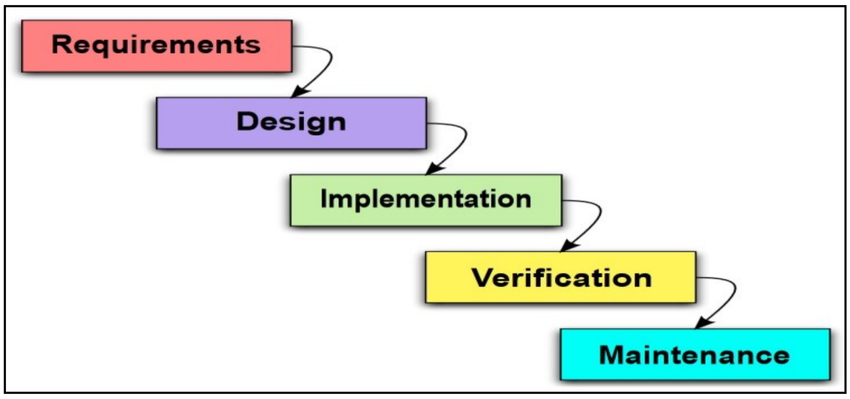


Figure 1.2: The Waterfall Model

1.3.2 Incremental Model

The incremental model divides the entire requirement into various builds (Figure 1.3). It consists of multiple development cycles. Cycles are divided into smaller, more easily managable modules. Each module passes through the requirements, design, implementation, and testing phases. A working version of the software is produced during the first module. Each subsequent one of the modules adds function to the previous release. The process continues till the complete system is achieved. Although it is flexible and generates working software quickly, its total cost is higher than the waterfall's. Also, it requires good planning and design and a clear definition of the entire system before it can be broken down and built incrementally. [6]

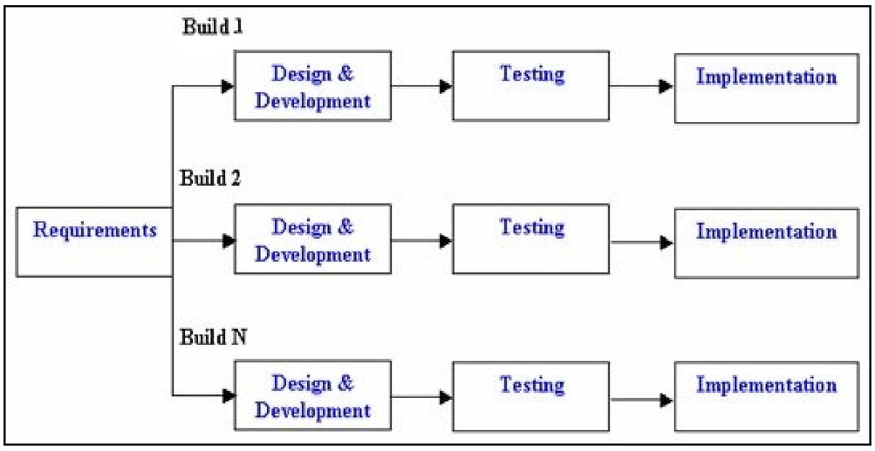


Figure 1.3: The Incremental Model

1.3.3 Spiral Model

The spiral model is similar to the incremental mode, with more emphasis placed on risk analysis. It consists of four phases, planning, risk analysis engineering and evaluation. The software process is presented as a spiral and it's based on iterations (Figure 1.4). Each subsequent spiral builds on the baseline spiral. The requirements are gathered during the planning phase. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. The model is best applicable to large projects whose cost is an important parameter, in contrast to smaller projects. It has a high amount of risk analysis, thus the avoidance of risk is enhanced. However, risk analysis requires a highly specific level of expertise and the project's success depends on the risk analysis. [7]

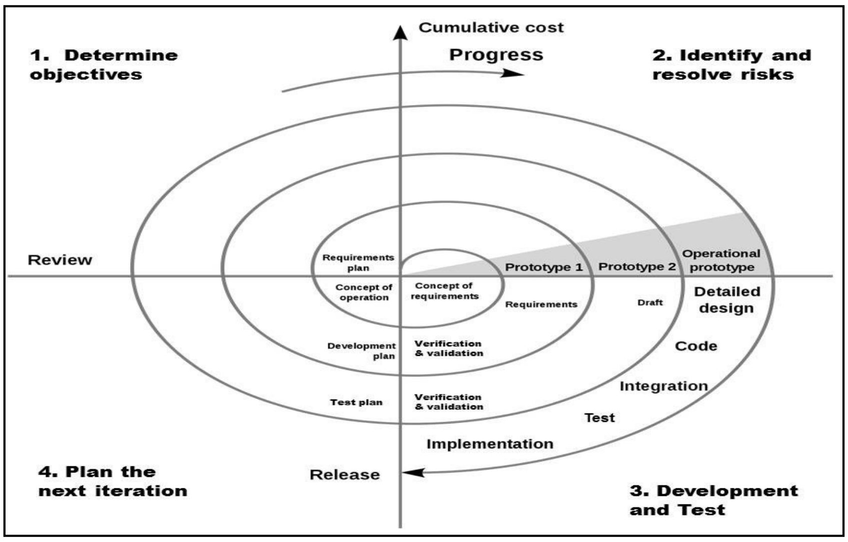


Figure 1.4: The Spiral Model

1.3.4 Agile Model

The Agile development model is also a type of incremental model. The software's development is done in incremental, rapid cycles, thus there are small incremental releases with each release building on previous functions. Each release is thoroughly tested to ensure software quality is maintained. The main advantages are that working software is delivered frequently and that even late changes in requirements are welcomed. It is characterized by regular adaptation to changing circumstances and continued attention to technical excellence and good design. However, in case of some software deliverables, especially the large ones, it is difficult to access the effort required at the beginning of the software development life cycle. There is lack of emphasis on necessary designing and documentation. Only senior programmers are capable of taking the kind of decisions required during the development process. Hence, it has no place for newbie programmers, unless combined with experienced resources. [8]

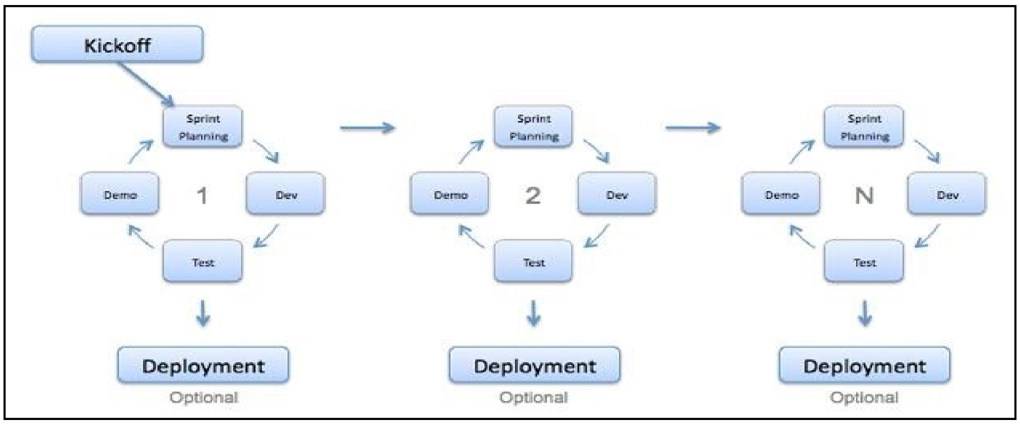


Figure 1.5: The Agile Model

1.3.5 Approach

The agile model was chosen as the appropriate approach for this project. As discussed in the previous section, the agile model is appropriate for senior developers which have experience with similar projects. It welcomes changes in requirements and provides frequent small incremental releases which are improved in the following ones. The current project implements an Android application, which requires a good design and technical excellence. In order to produce a successful application, many increments will be produced, each one of which is to be an improved version of the previous release. In each increment, there will be changes in requirements and improvements in design that will lead to new software development cycles.

The phases of the agile model are followed in this project. More specifically, the following phases will take place:

1. **Requirements specification:** A complete description of the behaviour of the system to be developed. Identify the requirements for the project development.
2. **Design:** Define the system's architecture and identify the important components of the system.
3. **Implementation:** Implement the system design into code.
4. **Evaluation and Testing:** Test that the system code has met its specification, evaluate the success of the previous phases and make modifications to them.

The use of the agile model also means that the order of the phases in chapters (next section) is not absolute. For example, new requirements needed during the design process may lead to going back to the requirements chapter and defining them.

1.4 Report Structure

The report contains seven chapters, excluding the introduction:

Chapter 2 – Literature Review: The literature review of m-learning, a study of the design features, principles and design patterns of a mobile application, the analysis of Android OS and a description of the needed tools to create the project's application are given.

Chapter 3 – Mobile App Design Guideline: This chapter gives guidelines for the design of a mobile application and provides tactics for solving problems in Android application development to help new developers.

Chapter 4 – Analysis: The requirements analysis of the application to be developed is given in this chapter.

Chapter 5 – Design: This chapter describes the design of the application.

Chapter 6 – Implementation: This chapter describes how the application is implemented.

Chapter 7 – Evaluation and Testing: Evaluation of the project which is based on functional testing, expert evaluation, the tests conducted on the software, findings and fixes are presented in this chapter.

Chapter 8 – Conclusion: Shows the main achievements of the project and proposed future work.

Chapter 2: Literature Review

In this chapter, the existing theories and technologies that have been considered and used in order to develop the mobile application are presented. Section 2.1 discusses the theory of mobile learning. Section 2.2 describes the design features, principles, and user interface design patterns for mobile application development. Section 2.3 describes the operating system which the mobile application will run on and the libraries that have been used for its implementation. Section 2.4 describes the Eclipse development environment which is used to implement the application.

2.1 M-learning

This section presents the theory of mobile learning that is related to the project. A first approach, the features, a mapping of the landscape, the perspectives and the benefits of m-learning are given in this section.

2.1.1 Learning through Mobile Devices

The special thing about learning through mobile devices (m-learning), over other types of learning, is that learners can be constantly in motion [9]. Therefore, an advantage of m-learning is that we have the ability to take ideas and learning resources from any place and to apply or develop them in another place, sometime thereafter. So, we can use the knowledge we gained at one time at a later time, or perhaps even in a different context of learning.

Also, we can move from one learning object to another, as well as manage a number of different learning objects, without necessarily following a uniform program of studies. We can manage our use of technology as we see fit. For example, we can choose when to use our mobile phone depending on our needs.

Learning through mobile devices does not try to separate learning from other forms of educational activity, as some aspects of informal learning and learning that takes place in workplaces are fully movable. Even students within a school will move from a class to another class and from a topic to another topic. [9]

So, what we are interested in learning through mobile devices is how knowledge and skills can be transferred, e.g. from home to school and vice versa, how we can manage learning through mobiles at different times of man's life and how new technologies can support a human society in which people are increasingly constantly on the move. People can learn through mobile devices in more and more situations of their daily life.

Studies show that, interest-wise, only 1% of learning occurs during someone's movement. It seems that existing technology cannot adequately support learning during someone's movement. So, there may be opportunities to design a new technology which can support learning during the ever-increasing time when people travel, or namely are "in motion". [10]

Also, more and more people, especially young people, possess one or more "smart" mobile devices. This should be particularly taken into consideration if we want to invest in a new learning generation, more innovative and modern, which will take mobile devices into account.

We should take into consideration that mobile phones give the opportunity to not only make phone calls, but offer additional benefits through their supported services, such as sending and receiving text (SMS) or multimedia messages (MMS). So now we see a convergence of mobile technologies, in functions that combine phone, camera and wireless multimedia computer. Another equally significant convergence occurs between new mobile technologies and new concepts of lifelong learning.

By most, learning is considered a collaborative process and activity that takes place anywhere people have to share knowledge. So, mobile network technology allows people to communicate regardless of their geographical location.

Computer technology in learning can be ubiquitous. People – users are now given the chance to maintain and organize their digital learning files for a lifetime and constantly carry those files with them. [11]

To summarise, there is a suggestion that the theory of m-learning must be tested against the following criteria: [9]

- Is it significantly different from current theories of classroom, workplace or lifelong learning?
- Does it account for the mobility of learners?
- Does it cover both formal and informal learning?
- Does it theorise learning as a constructive and social process?
- Does it analyse learning as a personal and situated activity mediated by technology?

2.1.2 Criteria of Learning through Mobile Devices

The criteria that should be considered are the following: [9]

- ***The learner is the one who moves and is "mobile" rather than the technology***: We should therefore not only focus on the design of specific portable technologies, but also on the interactions between learning and technology. Because they are complex and diverse, with students to be informed occasionally, through whatever technology is available, including mobile and fixed telephones, computers of other people, as well as learning resources from books and notebooks, so they should be taken into this particular account.

- ***Learning is merged into other activities as part of everyday life:*** So it cannot be separated easily from other daily activities such as conversation, reading, or watching TV. These activities can be resources and contexts for learning. Learning is integrated into processes that are not directly related to it, such as when doing shopping or enjoying entertainment. So it is organized into projects, which are connected with our daily activities, and whenever someone tries to solve a problem in everyday activity [12].

- ***Learning can generate as well as satisfy goals:*** A need or a problem that may arise from curiosity, as we randomly discover something while we move, can encourage us in a process of learning and it can form new goals that can be explored through formal or informal study**.**

- ***The control and management of learning can be distributed:*** In a classroom, the teacher and the learning locus, i.e. the classroom, remain firmly connected. But in learning through mobile devices, learning can be distributed among students, drivers – counsellors, teachers, technologies and various learning resources around the world, such as books etc.

- ***Context is constructed by learners through interaction:*** To explore the complexity of mobile learning it is necessary to understand the context in which it occurs. Context should not be regarded as simply what surrounds the pupil at a given time and place, but as a dynamic entity that is manufactured by the interactions between students and their environment.

- ***The theory of m-learning can both complement and conflict with formal education:*** So students can extend their classroom learning to travel, museum visits, etc. They can constantly replay their educational material on mobile devices or collect and analyse new data and information on them. They could also "disrupt" the carefully run environment in order to carry their mobile phones to record the lectures of the course, or enrich it with lectures heard in places outside the school [13].

- ***M-learning highlights and deep moral issues such as those of privacy and property:*** Systems that allow people to record their daily lives in sounds and images and then reuse them could draw attention to powerful tools for lifelong learning. They even allow parents or teachers to check the process of learning in detail, so the play and leisure can take place with the extension of school activity. This may be regarded as particularly troublesome for students.

2.1.3 Learning through Mobile Devices Features

The discrete aspects of learning through mobile devices: [9]

1. The **mobility**. It can take place in areas outside a typical classroom.

2. The **informal distribution** of participants. Participants may be distributed in different areas, even outdoors where the use of mobile devices is necessary.

3. The type of **interaction** between learning and mobile technology.

Of course, we must not consider learning through mobile devices as just another type of educational interaction, because we risk losing the broadest sense of the learning concept [14].

If we want to talk about the era of mobile technology, we can talk about an education that is based on continuous interaction through mobile technology. So there is a dialectical relationship between technology and active learning.

2.1.4 The Two Layers of M-Learning

It has been found through the analysis of activities using mobile learning that in an activity done with mobile devices, we have two perspectives, of tool-mediated activity [9].

1. ***The semiotic layer:*** In which the student internalizes the external knowledge as a private thought that provides the resource for the activity's testing and development [15].

2. ***The technological layer:*** This refers to learning as an interaction with technology, in which tools such as computers and mobile phones operate as interactive agents that help us communicate and learn through the recall of information.

These two layers can be studied separately to provide either a semantic framework to promote discussion with theorists of education and analyse learning in the era of mobile devices, or a technological framework for software developers and engineers that will explore the requirements for the design and evaluation of new mobile learning systems.

These two layers can also be studied together. We could describe technology as any tool that serves the purpose of research and allows us to transform problems into new knowledge [16]. Consequently, computers, languages and ideas can all be suitable as technologies for research and there is no clear distinction between the semiotic and the technological layer.

Learning occurs as a sociocultural system, in which many students interact to create a collective activity that is framed by cultural constraints and historical practices.

2.1.5 Valuable Elements and Benefits of M-Learning

The core of learning via mobile devices is that it allows students to be in the right place at the right time, so they can gain experience in an authentic learning environment, wherever that is.

M-learning is identified by three different elements that are all valuable to teachers and students who respectively teach or learn: [17]

- **Usability and convenience**
  Learners are able to utilize their time even when they are in motion. For example, on the bus or on the train, they can write or read their notes. We can say that learning through mobile devices refers to a society on the move.

- **Efficacy and expediency**
  Efficacy – expediency refers to:
  a. Students can be in one place and collaborate with others who are somewhere else, either via voice calls or SMS messaging.
  b. They can access the Internet from anywhere.
  c. They can take photo material which they will edit later.

- **Immediacy**
  Students can directly share their ideas with their partners and thus get immediate feedback. Generally we can say that mobile devices provide the opportunity for students to communicate with each other and with their teachers anywhere in the world. This way, they can receive the information and the knowledge directly when they need it, in order to solve specific problems, satisfying their curiosity.

Beyond the aforementioned most valuable elements, there are additional advantages to the use of m-learning: [18]

- **Convenience and flexibility:** it can be accessed anywhere and at any time.
- **Relevance:** it enables training to be situated rather than simulated, so it makes learning possible at the point of need.
- **Learner control:** the always-available nature of mobile learning empowers learners to take initiative and direct their own learning activities.
- **Good use of "dead time":** it can happen during "dead time", while travelling or waiting for a meeting to start.
- **Fits many different learning styles:** reading (text and graphics), video, animation, working through decision trees, listening to podcasts, contributing to discussions (forums or SMS), researching on the internet, choosing the correct answer (text or photograph), are all means of offering learning on mobile devices.
- **Improves social learning:** SMS texting reminders, knowledge sharing forums, "ask a question" forms and the use of telephony are all means that enable interaction between peers and tutors using mobile devices.
- **Encourages reflection:** the voice recorder on many mobile devices enables effortless and instantaneous recording of thoughts and opinions.
- **Easy evidence collection:** the portability of mobile devices makes them readily available to collect portfolio evidence via audio recorder, or video camera.
- **Supported decision making:** mobile devices offer timely access to information, which enables the quick double-checking of a decision, and thus better professional judgements.
- **Speedier remediation:** it enables forgotten or mistakenly remembered information to be speedily accessed and redressed.
- **Improved learner confidence:** short nuggets of learning offered on mobile devices, accessed prior to meetings or beginning tasks, improve learners' confidence in their skills.
- **Easily digestible learning:** the small screen minimises the amount of information that can be offered to a learner at any given time, and so avoids cognitive overload.

- **Heightened engagement:** quick-fire knowledge or mobile assessments/quizzes, in between other kinds of training activities, keep learning fresh and at the forefront of learners' minds, making success more likely.
- **Better planning for face-to-face sessions:** quick pre-assessments via mobile devices, prior to face-to-face sessions, enable trainers to determine learners' level of knowledge and plan their sessions accordingly.
- **Great for induction:** induction into mobile devices enables learning to be contextualised to the exact spot in the workplace it makes reference to.
- **Elimination of technological barriers:** the use of a learner's own mobile device means they are already familiar with the technology, eliminating technological barriers to accessing learning.
- **Designed once - delivered across multiple platforms:** having been developed using a mobile authoring tool it allows for a single design to be delivered across platforms to many different devices.
- **Easily track able via Wi-Fi:** it can be designed with an offline capability that enables tracking data to be saved if connectivity is lost, and then synchronised when a wireless connection is available again.
- **Cost-effective build:** it is cheaper than booking the resources required for face-to-face training or supplying laptops and other computing devices for e-learning. And it can be easily pushed out to learners' personal devices.
- **Direct interaction with learning:** with most mobile devices, the use of touch screens and other more direct input devices removes a layer of interactivity, meaning the learner literally is interacting with the learning.
- **Big data tracking:** with the integrated connection of mobile devices to the web, it opens up the possibility of tracking everything the users do, how they use the training, what questions they got right and even their behaviours.
- **Personalisation:** by getting the user to do the training on their own personal device, they are more likely to engage in learning. They are also more likely to do the training on their own time, rather than at work.

2.1.6 Current Perspectives of M-Learning

A mobile device could change the way we learn in practice. For example, when students know that everything is recorded or can be easily recorded, it changes their behaviour. Consequently, we must focus on activities and dialectical relationship between the student and the technology and not on people or technology separately.

The technology of mobile devices must take into account that the aim is not to just transfer material to a smaller screen but to create learning incentives, so that learning through devices does not become a victim of its success. The current prospects of learning through devices is generally classified into the following four broad categories: [19]

1. **The perspective of the used technology.** This perspective considers the nature of mobile devices that will be used for m-learning, such as a mobile phone, pads etc.

The main elements of m-learning are the mobile devices which are used, the wireless application protocol (WAP) [20], and the wireless applications (web, sites, portals). Together, they enable new forms of educational activities and learning services provided to users of mobile devices.

2. **The perspective of the relationship of m-learning with e-learning.**

This perspective characterizes the m-learning as an extension of e-learning.

3. **The perspective of strengthening of formal learning.**

Classic education is often defined as teaching face to face. However, it is not at all clear whether this view is completely correct. Various forms of distance learning (for example, correspondence) have existed for over 100 years [21], leading to questions about the position of m-learning in relation to all forms of traditional learning, even learning which was happening by mail, from a distance and not only in the classroom.

4. **The perspective that examines m-learning through the prism of the student.**

An initial statement connected m-learning with the potentiality and convenience which mobile devices support for lifelong learning [22]. However, it soon became clear that we should focus on the mobility which is offered by these devices to the same student.

Thus, m-learning can be considered as any learning which happens when a student is not in a specific place, but is moving. This perspective focuses on learning through activity [23].

2.1.7 Key Features of M-Learning

There is a general agreement that a precise definition of mobile learning is unattainable. Instead, the main characteristics of m-learning are the following: [24][25]

- It allows students to build their knowledge in different learning contexts.
- It allows learners to understand the object of learning by themselves.
- Mobile technology can often change the plan of the learning activity which should be executed.
- The mobile learning contexts are about more than time and space

The framework of mobile learning is further beyond time and space. All in all, the role of m-learning lies in how to guide the student into forming his own opinion.

2.1.8 What Factors are Creating Learning Incentives in M-learning

There are five main factors in creating learning incentives in m-learning: [25]

1. The **control** of learning objectives which have been set. As such, students find informal learning through mobile devices more interesting, because they feel free to do extra activities and actions related to the subject that they are studying. Thus, they have the control of the learning purposes, something that they cannot easily do in a typical classroom during the formal learning process. Also, through the use of mobile devices, it is possible for external incentives, such as to get a good grade in the course, to be converted into internal incentives, while the use of technology increases the students' involvement in more interesting activities, which cannot take place in a hall.

2. **The sense of ownership** and **autonomy** for the holder of a mobile device. Thus, the same device can be used differently by different users.

The rate of learning's creation incentives refers to how students use and adjust their mobile devices to various activities, but also to how it is possible for the same mobile device to change and adjust the activity which the student must do, providing him with extra capabilities. But this adaptability of technology can be both subversive and used in ways that were not originally foreseen by its designers, and it can eventually undermine the original purpose.

3. **The fun**. Having the appropriate mobile device and its usage for the appropriate activities create a more "special" identity, especially in the young ages. This can be fun and creates its own motivation for learning at these ages. A device like that, with graphics, sound, images etc. triggers the_curiosity to learn while you have fun.

4. **The communication**. A mobile device allows users to share various things ranging from blogs to photographs. This trend of sharing any information can itself be an incentive to learn, as can be seen from the number of sites that support social networking.

The current plethora of open software and free access to such sites can be seen as an incentive to share information and thus to create incentives for learning.

However, it was noticed that this is often a one-way process. People wish to offer such learning resources as photos, or their experience on a subject, but they are less concerned about what others want to share with them.

Also, unlike personal computers, mobile devices can provide instant and direct access as they can always be activated. This in itself is an extra incentive to learn.

If students hold a mobile device, it allows them to set questions and answers in an online conversation, even during a lesson. Students usually find this supportive to learning. It is also suggested that using a mobile device can be even more useful to students with lack of confidence, who are less likely to participate in a face to face discussion. It seems that it is easier for students to do this, when they have the opportunity to participate in a discussion in a more private manner, which is supported by mobile devices. The use of mobile devices gives more emphasis on the interaction between the students, as it seems to support collaborative learning activities.

5. **The confidence.** As reported [26], the use of mobile devices develops a sense of confidence, which is connected to the fact that the owners of mobile devices can access

the Internet wherever they are. Search engines are very useful in creating self-confidence and enable users to easily find and directly answer a question posed by the teacher. Students report that they feel very well when they respond to questions from their teachers, something that could not have been done without the use of a mobile device.

2.1.9 Management of Conflicts between M-learning and Formal Learning

The question is how we can manage the conflict between personal informal learning through mobile devices and formal learning that takes place in a typical classroom. [27]

Most schools and colleges still do not recognize the informal networked interaction as a legitimate form of learning and they forbid students to bring their phones and personal computers in the classroom.

Consequently, there are systematic tensions between technologies and the educational system's activity concerning informal education through online social learning through computers networking, which takes place in a classroom.

Soon, to a certain extent, these tensions could be converted into conflict, since informal activities through mobile devices could be seen as more interesting by the students. This problem may become more pronounced if schools do not quickly adapt to the changing world's interaction with networking through mobile devices. [27]

2.1.10 Arguments For/Against the Use of Mobile Devices in the Classroom

There are specific arguments for the use of mobile devices in the classroom: [25] [27]

- The technology for social networking is growing rapidly and young people are increasingly using this technology.
- The need of communication fits with the culture of the youth.
- Teenage culture is open to sharing knowledge and to collaboration through communication, although schools insist on traditional teaching.
- Schools should incorporate personal – mobile technology that children can use in an evolutionary manner, for the transition to new ways of learning to not be persistent.

Of course there is a systemic reaction by schools and also by parents. Most of them face the new technology as a threat and a danger to children. So there are arguments against the use of mobile devices: [25] [27]

- Schools are valuable for their assistance to students, and this cannot be changed in the near future.

- Students are able to distinguish between informal and formal learning, and they understand the need for a clear separation between the classroom and informal learning that can take place strictly and exclusively only outside of class.

- Some forms of knowledge are not accessible without an official pedagogic process in which the role of the teacher is crucial. Informal social networking develops certain skills but it cannot replace formal learning.

- Education has a long history and online social networking should not change school practices. New technologies and practices should be integrated into education in ways that will take advantage of their power, but that will not allow the division of schools.

Summarizing the above arguments, we can consider that the future scenario is that of schools that neither welcome nor prohibit mobile technologies and online social networking, but rather adapt to new technologies and their opportunities.

Schools can allow students to have mobile devices, but to use them in classrooms under controlled conditions and with regulated access to networks.

Even students should learn how to adjust their networking practises to the school environment, regulating the interactions between informal and formal learning, in order to gain from the use of mobile devices.

2.2 Design Features for Mobile Application Development

This section presents the design features that would be beneficial for a mobile application to have. A description of the design characteristics of m-learning applications is given, followed by an analysis of the general mobile design principles and mobile user interface patterns.

2.2.1 Design Characteristics for M-learning Applications' Activities

Some key points for the proper design of the activities that are supported by the mobile devices are the following: [28][30]

- The relationship between the student and the teacher must be important, because it helps students understand the technology and apply it to practical educational scenarios in basic education.

- The various activities that rely on mobile devices should help teachers associate the use of mobile devices with traditional teaching in classroom.

- The moral dimension is critically important as we move towards a world where technology is ubiquitous. So we need to respect the privacy of each student.

- The representation of data on mobile devices is also an issue. What must be taken into consideration is not only the small size of the screen, but what representations are more suitable for data presentation of the course. For example, poor presentation of the text on the screens of mobile devices acts as a constraint on the optical types.

- The role of mobile devices in the process of socialization and its impact on learning is still under investigation.

The important thing is to see that m-learning can play a more facilitating role in learning, rather than its own autonomous role.

2.2.2 Mobile Design Principles

Mobile devices differ from computers in many ways: [30]

- Tiny screen
- Battery powered
- Spotty connection
- Small pipe
- Expensive data
- Limited storage
- Distracted user
- Touch vs. mouse
- Personal
- Always on
- Always with
- Usually connected
- Directly Addressable
- GPS
- Gyroscope
- Accelerometer

We need to consider a few important things when we want to design a new mobile application or to transfer a full-sized computer application to a mobile environment: [31]

- ***Small Device*:** The small screens of mobile devices cannot support multiple windows and only smaller sizes of layered information can be used, such as drop-down menus, pop-up menus, and small dialog boxes.

  Small screens also prevent users from reading large texts because the context can be easily lost while moving. Thus, the content of a mobile application should be carefully designed for a small screen and in a way which will help users focus while reading.

  Users can interact with only one application at a time. Thus, every application must be characterized by responsiveness and provide its functionalities very fast, so it should pre-fetch data whenever is possible, to decrease the delay time.

- ***Personal Device*:** As a mobile device is a personal device, an application should be designed to hide any password entry or secure input data. It is easier for a user to hide the screen rather than hide the device's keyboard.

  An application should provide options to store private information, such as login details, to allow faster usage without the need of re-typing the entry data. Also, if the application is using cookies, it should not clear them quickly, as nobody can have access to it when the device's network is lost.

- ***Customized Device*:** An application should be attractive to users and provide the option to use different themes or background wallpapers. Some users may not like some colours or they may have difficulties in reading some text sizes and thus options to customize the user interface should be a good feature.

- ***Always On, Always Connected:*** Users want to use their mobile phones even if they are in places where mobiles aren't permitted. Thus, the application's design should provide silent or vibrate modes, to allow users to use them without being noticed by others.

- ***Battery-Powered*:** Mobile devices use batteries as their power source and its processor power, screen display, Wi-Fi and network connectivity increase the power usage. An application must be designed to use less power, such as pre-fetching data while the screen is off. It must allow users to choose how often its services will run in the background and notify them about the power demand of each choice.

- ***Inconsistent Connectivity*:** An application needs to be designed to handle inconsistent connectivity gracefully. If the application contains infrequently changing data to which the user needs reliable access, pre-fetching data can ensure that the user will have the available data when he asks for it, regardless of the availability of a network at that time.

- ***Difficult Text Entry:*** Text entry is more difficult than on full-sized computers. Users prefer to type short texts while they are using a mobile device. An application should provide other input sources, such as cameras, address books, calendars, auto-completion, speech, image recognition and global positioning system (GPS).

- ***Handling Device Proliferation:*** Handling device proliferation is a necessity. An application should run on multiple devices. To accomplish this a mobile designer must:

    - ➢ Select a set of targeted mobile devices and write the application that works on them.

    - ➢ Select technologies and designs that work on all devices.

    - ➢ Use technologies that convert the basic core functions into the format needed by each selected device.

    - ➢ Separate the devices into groups based on the common characteristics, design the core functions for each class separately, and then make the necessary changes for each device using the available automatic tools.

There are also more things that always matter:

- **Informative** – is extremely valuable
- **Targets** – must be finger friendly
- **Controls** – should be beneath content
- **Scrolling** – should be avoided when possible
- **Tab Bar** – if exists, three to six major categories
- **User Input –** avoid autocorrect
- **Typing** – avoid often typing
- **Keyboards** – should be apropos for each field
- **Support Landscape** – if the application requests lots of typing
- **Visibility** – avoid hiding options
- **Portrait –** optimize portrait layout first
- **Disorientation** – handle anything that can cause disorientation
- **Feedback –** provide feedback for every user interaction
- **Modal Alerts –** use only for big issues
- **Confirmations** – use for actions
- **Badges** – good to use
- **Resume Operation** – when user re-opens the application or after screen is turned off
- **Launch Screen** – should be “content-less” background
- **Icon** – use icon for the application, it is like a business card for it

An analysis with guidelines about the above things is given, followed by solutions that work in a wide range of mobile applications, in Chapter 3.

2.3 Android

This section presents the mobile software which our application will run on. An analysis of the Android system and its features is given, followed by the versions of Android and the needed development tools to create an Android application.

2.3.1 What is Android?

The name Android is coined from the Greek word andr-, the meaning of which is "man", and the suffix -oid, the meaning of which is "of the species", so Android means as much as "being human" [32].

Android is a complete, open and free platform [33], designed primarily for touch screen mobile phones, such as smart phones and tablet computers. It was founded way back in 2003, it was developed by Andy Rubin, Rich Miner, Nick Sears and Chris White, in Palo Alto, California and it was purchased by Google in August, 2005.

Android includes an operating system which is based on the free Linux kernel, the necessary middle-ware, libraries and key mobile applications. The diagram in Figure 2.1 shows the major components of Android, which is divided into four different layers that include five different groups.

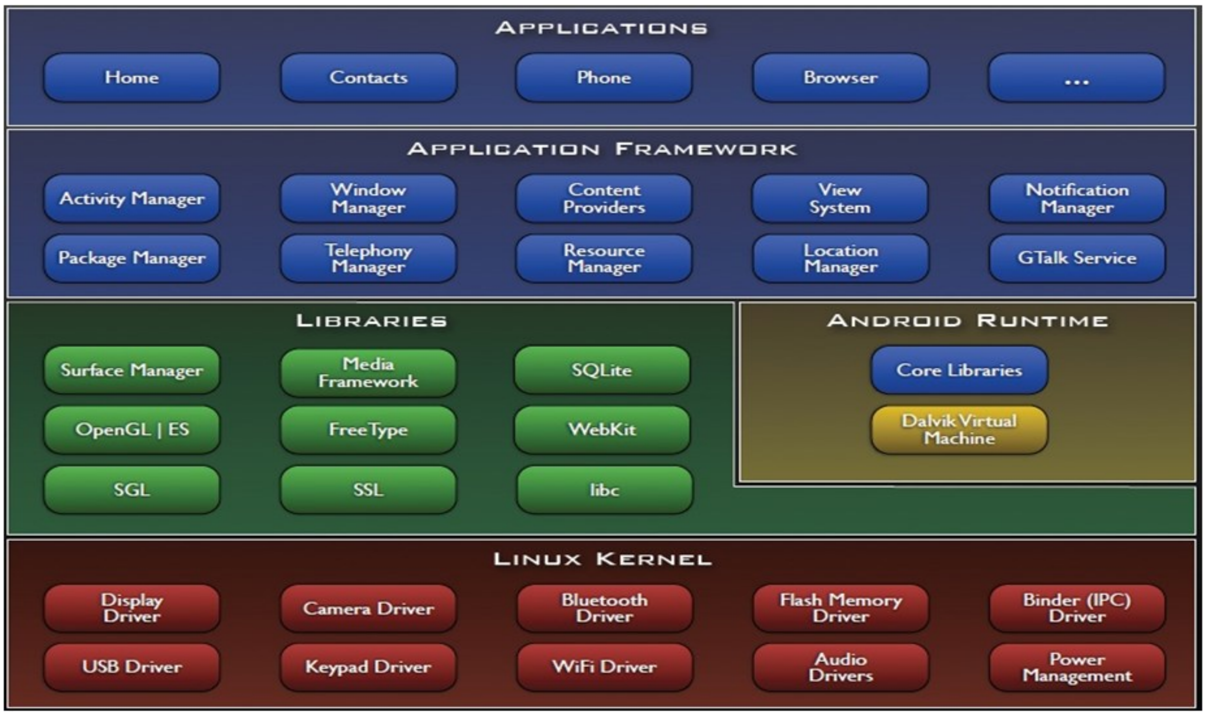


Figure 2.1: Android Architecture

A description of the most important of Android's components is given below: [32][34]

Linux kernel – Core services, including the hardware drivers, process and memory management, security network and power management are handled by a Linux 2.6 kernel. The kernel also provides an abstraction layer between the hardware and the remainder of the Android architecture stack.

Libraries – Running on top of the kernel. The available libraries are all written in C/C++. The core libraries are the following:

- **Surface manager** – provides display management
- **Media Framework** – A media library for playback of audio and video media
- **SQLite** – provides database support
- **OpenGL | ES** – graphics libraries for 2D and 3D graphics
- **FreeType** – provides font-related operations support
- **WebKit** – integrated web browser and Internet security
- **SGL** – graphics libraries
- **SSL –** provides Internet and web browser security
- **libc –** support for Android-specific services such as system properties and logging

Android Runtime – The runtime is what makes Android something more than a mobile Linux implementation. Android runtime is the engine that powers the applications and, along with the libraries, forms the basis for the application framework. It also includes:

- **Core Libraries –** provide most of the functionality available in the core Java libraries, as well as the Android-specific libraries.
- **Dalvik VM –** is a register-based Virtual machine that has been optimized to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low level memory management.

Application Framework – provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.

Applications – All applications, both native and third-party, are built on the application layer by means of the same API libraries. The application layer runs within the Android runtime, using classes and services made available by the application framework.

As Android is a multitasking platform, it can simultaneously run more than one application without one affecting the performance of the other. Android is an open source platform and thus it allows device manufacturers or third-party developers to modify it. This ensures that it evolves, continuously progresses and keeps pace with the latest technologies and developments.

Android has seen a number of updates since its original release. These updates fix bugs, add new features and improve the performance of the operation system. After the alpha, beta, and first version of Android, it has been decided by Google to use code names based on dessert items in alphabetical order for each new version, such as Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich and the current one Jelly Bean.

2.3.2 Android SDK

The applications developed by third-party developers have no difference from the existing device's applications. They might as well have access to all the main features of the device, which allows users to fully enjoy its potential. For example, users can use applications to display songs or videos that are stored on their device.

The development of applications is not limited to the above, but an application can have access to the web, to transmit data between the device and the web and display them on the device's screen.

Google's Android Software Development Kit (SDK) [36] and the use of the Java programming language are necessary to the development of an Android application. The Android SDK includes everything needed to develop, test and debug an Android application. A description of the included components is listed below: [35]

- **Android APIs** – They consist of the core of SDK and they provide access to the Android stack.
- **Development Tools** – SDK includes tools that let a programmer compile, run, and debug applications.
- **Android Emulator** – It is a full interactive Android device emulator. Running the applications on the emulator is the same as running them on a real Android device. It also provides different interfaces and options to run the application on a specific device emulator or with custom hardware features (RAM, CPU, and Resolution).
- **Full Documentation** – It includes extensive code-level reference information detailing exactly what things are included in each package and class and how to use them. It explains how to get started and gives detailed explanations of the fundamentals behind Android development.
- **Sample Code** – SDK includes a selection of sample applications to help understand some fundamental Android APIs and coding practises. Each version of the Android platform available by the SDK Manager [37] offers its own set of sample apps.

SDK can be installed in computers running the latest distributions of Linux, Mac OS X, and Windows, so it makes developing free in almost all platforms. It also provides older versions of the Android platform, in case developers wish to develop and run applications on older devices.

2.4 Eclipse in Android Development

This section presents the Eclipse environment and its plug-in, which is used to develop an Android application.

2.4.1 What is Eclipse?

Eclipse is a free multi-language integrated development environment (IDE) [38]. It is written mostly in Java and can be used to develop applications in Java and other programming languages, such as Ada, C, C++, etc. It is one of the suggested development environments both in academia and industry.

Eclipse started from IBM, including a group of industry leaders who formed the first Eclipse consortium in November 2001. After three years, in 2004, IBM released Eclipse as an open-source platform. After that, all Eclipse projects are released under the Eclipse Public License (EPL). [39]

The value of the platform is what it encourages: rapid development of integrated features based on a plug-in model. In order to add a new component in Eclipse, a new plug-in is needed. An Eclipse plug-in is another Java program, which provides a new functiona within the context of Eclipse’s environment. There isn’t any limitation as Eclipse can accept hundreds of plug-ins through the plug-in manager, which also supports an updater service. [40]

2.4.2 ADT Eclipse Plug-in

Eclipse is the first choice for a developer to start developing an Android application. Eclipse is a cross platform environment and thus it can run with SDK in almost all platforms. It needs the Android Development Tools (ADT) plug-in [41] to use the Android SDK. ADT is designed to provide a powerful, integrated environment in which a developer can build Android applications.

Eclipse, using the ADT, can create new Android projects and application user interfaces, add packages based on the Android Framework API, debug, compile, and run the applications using the SDK tools [41]. It also provides the SDK Manager, on which developers can download all versions of Android and any needed libraries.

ADT also gives you access to other Android development tools from inside the Eclipse IDE. For example, it lets you access the many capabilities of the DDMS tool [42] (take screenshots, manage port-forwarding, set breakpoints and view thread and process information directly from Eclipse). It automates and simplifies the process of building an Android application. It provides an Android code editor that helps in writing valid XML for the Android resource files.

Lastly, Android applications need to be installed in mobile devices and Eclipse with ADT provide the export manager that packs an Android application project to .apk format. Then developers can easily share the .apk files to users for the application’s installation on Google Market or any third party Android stores.

Chapter 3: Mobile App Design Guideline

This chapter provides the principles of mobile interface design and recommendations on how a mobile application developer can follow these principles, followed by the mobile user interface design patterns and their solutions to mobile application design, and tactics for solving problems in Android application development.

3.1 Principles of Mobile Interface Design

In Chapter 2, it has been mentioned that mobile devices are different from personal computers. Thus, the designer needs to follow specific mobile interface design principles and be aware of the fact that there are also other things that matter. From our experience in using and developing mobile applications and from a literature review [43], the below principles are useful for mobile application developers.

- ***Informative:*** It can be combined with responsiveness and it is absolutely critical. Whatever the application does, it must inform the user, using an alert dialog. Thus, when the user requests an operation that takes a lot of time, he is informed instantly by the alert dialog and that way the application also retains its responsiveness quality.

- ***Targets:*** Everything that needs to be interactive should be finger friendly. The user must be able to easily press everything with one press, using his fingers. All the elements of the application's screen must be tested so that they are touchable by a finger. If an element is difficult to be pressed on, its size should be increased.

- ***Controls:*** Controls should be beneath content, because users used to keep their fingers on the bottom of their mobile device, near the device control buttons. If the controls are used beneath the display, users will hide the content with their fingers. Also, if their mobile device has a big screen and the controls are not near the device control buttons, the user needs to move his fingers from top to bottom and vice versa, something that slows him down.

- ***Scrolling:*** It should be avoided when possible. As the device screen is not big, scrolling may hide content that the user wants to see at all times. Also, while scrolling, the content may be lost easily, as the scrolling interaction is really fast. Scrolling should be used when the application is using lists.

- ***Tab Bar:*** They can help users navigate the application more easily, but they should be between three and six. If they are less than three, it is better for another layout to be used. More than six tabs will lead to small tabs and this can make viewing and pressing on them more difficult.

- ***User Input:*** Autocorrect should be avoided in the user's input as it can be frustrating. However, an option of enabling/disabling autocorrect is acceptable.

- ***Typing:*** It should be used only when it is indispensable. Otherwise, it should be avoided, as typing demands the user's focus on the screen and it slows him down.

- ***Keyboards:*** They should be available in all text fields and they must not hide the text fields' context.

- ***Support Landscape:*** If the application requires a lot of typing it should support landscape orientation. This way, the user can use a larger keyboard and see more content on his screen.

- ***Visibility:*** The application should provide all available options in visible positions on the screen which are noticeable.

- ***Portrait:*** It should be optimized first as it is the default orientation of a mobile device.

- ***Disorientation:*** The application should be tested on all orientations and all elements must be checked to see if they are displayed correctly on each orientation.

- ***Feedback:*** It should provide feedback for every user interaction, otherwise the user will probably think that the action has not been completed. Feedback can be provided by alert dialogs, but a combination of alter dialogs, vibration, and led notification is preferable.

- ***Modal Alerts:*** They should be used only for vital issues, such as non-responding processes, loss of data, resource starvation, etc.

- ***Confirmations:*** Confirmation dialogs should be provided to every one of the user's actions that changes (deletes, inserts, updates) data and to actions that can stop other operations or exit users from the current application screen or session.

- ***Badges:*** They should be used for buttons or hyperlinks as they are easier to be noticed by the user.

- ***Resume Operation:*** When the user re-opens the application or after the screen has turned off, the application should show the last operation. Also, the operations must not stop while the screen is turned off but should run in the background instead. If the user doesn't want to have background processes, he can select this option in his device settings.

- ***Launch Screen:*** The application should have a launch screen, as it is easier to add background processes to run while the launch screen is loading and it is the best way to attract the user from the start. It should have a "content-less" background and an animation should be used instead.

- ***Icon*:** An icon for the application should be used because it is like a business card for it. The icon can be a small banner with the name of the application or it can be a picture that refers to the application's theme.

- ***Theme*:** The theme of the application should contain colours that can be easily seen by all users. Contrast colours should be used to create a learning-friendly and accessible environment. The same theme should be used in all applications screens.

- ***Text***: When you want your text to be flexible, based on the user preferences, define text font sizes using SP (scalable point) units. When it comes to text sizes, you will want to use density-independent units like DP (device-independent pixels) and SP. The SP unit is perfect for text sizes, as it is sensitive to the user's display settings.

Mobile applications design could be called a sensitive design, because it requires of a designer not only to focus on the user, but also on the device, since the design must be used as a combination of users' preferences and what the device can really do.

3.2 Mobile Design Patterns can be the Solution

Mobile application design is difficult and it may cause trouble to a new developer, thus mobile design patterns can be the answer to solving the common user interface design problems. The mobile application design patterns and the ways they can help us are given below: [44]

- **Screen** – There are layouts that can be used for the application's screens.
  - ➢ **List Layout:** It is used when you want to create a list of scrollable items. It should be used only for lists.
  - ➢ **Linear Layout:** It can be used for both vertical and horizontal orientation. Its elements are stacked one after the other. It can be used for login, register, and generally input details.
  - ➢ **Grid Layout:** It is a grid of items and it can be used for an album, music or video screen.
  - ➢ **Relative Layout:** One of the best layouts to be used on application design. It can group different layouts and it can be used for every screen design.
  - ➢ **Merge:** It is not a layout in itself but it consists of other layouts. It is used when the application requires having more layouts in one screen, so some of them are disabled during a specific process. An example is when a login form that is hidden when the user is pressing the log in button is replaced by the hidden relative layout that contains a loader.

- **Navigation types** – Good navigation in applications makes it easier for users to accomplish any task.
  - ➢ **List:** Use a list with each row to be interactive and transfer the user to another screen.
  - ➢ **Tab:** Use tabs on the top of the screen and while the user is pressing on them, have the area below change respectively.
  - ➢ **Springboard:** It is the default navigation type. The navigation items are placed in a grid and each one transfers the user to another screen.
  - ➢ **Gallery:** It brings to the surface pieces of content for navigation. The content can be individual articles, photos, slideshow or videos.

- **Dashboard:** It provides a roll-up of key performance indicators. It is useful for financial applications, analytics tools and sales and marketing applications.
- **Metaphor:** It is characterized by a landing page modelled to reflect the application's metaphor. It is used in games but it can also be used in applications to help people categorize their items (notes, songs, videos), in catalogues.
- **Mega Menu:** It consists of a big overlay panel with custom formatting and grouping of the menu options. It can be used for store applications, to browse, shop, sell and manage the application activities.
- **Page Carousel:** It is used to navigate through a set of pages using the flick gesture. The page indicator displays how many pages there are in the carousel; flicking displays the next page.
- **Image Carousel:** It can be a 2D carousel or one with more dimensions. It is used to display featured products such as movies in a cinema's application.

- **Invitations**
  - **Dialog:** Add a dialog to give instructions to the user or to ask him for confirmation. It should be "content-less" with icons on it, showing the content type of the dialog (error, notification, note, confirmation, etc.).
  - **Tip:** Add a tip to guide the user through each operation. The tip can be added anywhere in the application. It can be a small coloured area on the screen.
  - **Tour:** A tour can be the best guide for a new user that is trying the application. A tour should highlight the key features of the application. It should have navigation arrows, for the user to be able to move between the tour's screens. It should also contain images or videos showing the operations because users prefer watching to reading.
  - **Video Demo:** One of the best types of invitations for applications that reply to specific actions/interactions, since it demonstrates the application in action.
- **Sort** – You must choose a sort type for your displaying results.
  - **Onscreen:** Use tabs with few options of sorting on the screen and have them display the sorting results below them.
  - **Selector:** Use dialogs and display the sorting options. Allow users to choose one of the sorting options and after the choice is made, have the system display the results.
  - **Form:** The same as selector, the only difference being that the user must press on the existing confirmation button after choosing the sorting option in order to see the results.
- **Search** – Searching for data using keywords.
  - **Explicit:** Add a search bar and a search button on the top of the screen and show the results in the area below the search bar.

- **Auto Complete:** As with explicit, only as the user is typing, the area below the search bar will immediately show a set of possible results.
- **Form:** Use a search form with extra options for the user to enter more than one search criteria.
- **Scoped:** It is easier and faster to add the search criteria by scoping them before performing the search. While adding text to the search bar, a pop-up window below the bar is shown with the search criteria.
- **Dynamic:** Entering search text will dynamically filter the data on the screen.
- **Recent:** Entering search text will display the user's recent searches.
- **Results / View Results:** When a search is performed, the results can be displayed on the same screen or a dedicated screen. The results can be displayed in a table or list with filter options.

- **Filter:** Large data in an application require additional filtering.
  - **Dialog:** Use dialogs to display a pop-up window requesting the user to choose the preferred filter.
  - **Form:** Use a form with filtering options, and allow the user to choose by ticking them.
  - **Onscreen:** Use tabs as options for filtering the display items.
- **Forms** – Applications require forms for data entry and configuration.
  - **Sign In:** It should have a minimal amount of inputs such as username, password and option to register.
  - **Registration:** It should have a minimal amount of inputs such as username, password and retype password.
  - **Search Form:** It should have multiple inputs, or criteria to generate results.
  - **Multi-Step:** Forms that cannot be displayed on one screen can be displayed on multi-step forms. Each step should show the current step and the total steps flow.
- **Feedback**
  - **Feedback:** Provide appropriate and clear feedback to the user, so that he sees the results of his actions and knows what is going on with the system. Feedback can vary from simple progress indicators and confirmation messages, to more sophisticated animations and effects.
  - **Error Messages:** They should be displayed in plain language, explaining the problem and suggesting a solution.
  - **Confirmation:** It should be provided when an action is required and provide feedback in the confirmation box.

- **System Status:** Show the status of the system. For example, "loading data" as a displayed text, or a loader, instead of leaving the user to wonder if the application is frozen.

The above patterns could be used every time a developer decides to design a new mobile application. In the coming future, more features will be available to mobile developers, thus more patterns will be created to solve new problems.

3.3 Development Tactics

Mobile application developers are facing problems during application development. In this section, we focus on Android development tactics (solutions) that a developer can follow to solve known problems and deal with challenges that spring from the plethora of devices available on Android OS, as well as inconsistent operating system upgrades. The development tactics are given below: [45]

Tactics for solving Software Fragmentation

Problem Description: The Android operating system has several versions that run on different devices. Timely upgrades change the version of the OS on the device, which means that developers cannot just focus on the most recent versions of the OS; not everyone has upgraded.

Workaround: Learn which OSs are most popular and develop with the latest widely adopted version in mind. If you want to develop an application that will be compatible with more than one version of the operating system, use the min and max SDK version values (API levels) that are provided by the Android SDK. Thus, the SDK will not allow you to implement something that is not acceptable in that range of min and max SDK version.

Tactics for solving Lack of Software/Hardware Integration

Problem Description: As mentioned above, the Android OS is running on many devices. Thus, there is a variety of button configurations. For example, there are devices which do not have a menu button, and thus, developing a menu context will not work for them.

Workaround: Once you understand which devices your users prefer and how they use features like touchscreens and keyboards, you can start designing an application that operates intuitively for most of your users. Alternatively, you can check programmatically which device is running your application, thus you can implement different functions for each device.

Tactics for Fixing Application after Release

Problem Description: In case of finding bugs in your application, it is a must to update it on the Google Store. However, there are users that do not update their applications from the Google store, thus it is possible for users to gain a benefit from the application's bug.

Workaround: Implement an update service in your application which will have access to a web address that will provide the new update.

Tactics for decreasing the user interface layouts multitude

Problem Description: It is common for new developers to create many layouts, using different dimensions in their objects for each screen density, but this is time and space-consuming.

Workaround: Using the layout's margins (such as marginTop, marginLeft, marginRight, and marginBottom) you can define the space (in dp Android units) of each object from the screen sides. In Android, DP is an abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi (dots per inch) screen, on which 1dp is roughly equal to 1pixel. When running on a higher density screen, the number of pixels used to draw 1dp is scaled up by a factor appropriate to the screen's dpi. Likewise, when on a lower density screen, the number of pixels used for 1dp is scaled down. The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion. Using dp units is a simple solution to make the objects' dimensions in your layout resize properly for different screen densities. In other words, it provides consistency for the real-world sizes of your UI elements across different devices, hence there is no need for extra layouts.

Tactics for managing multiple UI Layouts

Problem Description: During the user interface design, it is needed to work with more layouts in one screen. This causes trouble to developers that are trying to work with objects of a layout, because when you declare more layouts in one file, it is not possible to change the objects' properties (such as height, width, colour, position).

Workaround: Create temp layouts and work on their objects separately. Then, merge them in one file, adding one layout inside the other.

Tactics for Project Building errors in Eclipse

Problem Description:

- missing source folder: "gen"
- project could not be built until build path errors are resolved
- unable to open class file R.java

Workaround: If you want to solve any of these errors, go to the project menu, select the project and then press clean.

Tactics for fixing Application's stop error

Problem Description: During the start/execution of your application, your application stops and log chat shows the message: android.content.ActivityNotFoundException

Workaround: Check if the activity is declared in AndroidManifest.xml file.

Tactics for fixing Eclipse R.java error

Problem Description: Eclipse can display an error that a file cannot be found. For example: R.layout.login cannot be found.

Workaround: Remove the android.R import from your source code.

Chapter 4: Analysis

This chapter provides the analysis of the requirements of the system under development. At first it provides the stakeholders and their roles in the application development, followed by the system requirements and use case diagram and scenarios.

4.1 Stakeholders

The requirements analysis demands to specify who the main stakeholders are. These people have direct or indirect interest in the system [4]. Below, the interest parties and a description of them are presented:

- **Primary users** – Ones who are going to use the application. They can be categorized into two groups:
    - **Owners** of the courses' groups (professors, teachers)
    - **Participants** of the courses' groups (students)

- **Mobile application developers –** Mobile application developers who work in the area of M-Learning and distance learning may have interest in the application, either to implement new functionalities in the current application or to use it as a pattern for their projects**.**

In terms of this project, the above stakeholders have the same interest in the application, to use a fully-functional application. Thus, they can be defined as one stakeholder, the user.

4.2 System Requirements

In this section, the system requirements are presented. The application requirements can be divided into Functional and Non-Functional requirements. Functional requirements define the capabilities and functions that a system must be able to perform successfully. Non-Functional requirements define the qualities and criteria that can be used to judge the operation of a system. [4]

4.2.1 Functional Requirements

A functional requirement defines a function or a component of a system. Functional requirements are supported by non-functional requirements, which impose constraints on the design or implementation. Generally, functional requirements are expressed in the form "system must do <requirement>". [4]

The format chosen to present the functional requirements is the following:

ID: *The identification number of the requirement.* **Description:** *The requirement.* **Explanation:** *An explanation of the requirement especially needed for those which are complex.*

The system's functional requirements are given below:

ID	Description	Explanation
FR01	Users must be able to sign up.	The user must be able to sign up filling out a form with his name, email and password.
FR02	Users must be able to log into the system.	Users must be able to log into the system using their email and password.
FR03	Users must be able to log out of the system at any time.	Users must be able to log out of the system at any time pressing the log out button in the application's menu.
FR04	The system must provide an error recognition message in case of error during the log in or registration process.	The system must show an error message to the user, highlighting the error, if any of the input details of the login/registration form are wrong.
FR05	Users must be able to change their email or password after they are logged in.	Users must be able to change their email or password after they are logged in adding their old email/password details as well as the new.
FR06	The system must provide an error recognition message in case of error during the detail-changing process.	If old details are wrong or new ones are wrong (empty fields), a recognition message is displayed to the user.
FR07	Users must be able to create a new course.	The system should provide a new course button in the menu of the courses' list and after that, a form, requesting name, department and institution.
FR08	System must provide users' courses in a list.	Users should be able to see their courses in a list. In that list the courses which they created or attended should be displayed.
FR09	Users must be able to add other users to their course.	The owner of a course should be able to add other users by adding their email to the provided form.
FR10	Users must be able to remove a course.	The system should provide a remove button while a user is pressing on a course. If the user is the owner, the system deletes the course from him and all the course members. If the user is a member the system removes the course only from this user.
FR11	The owner of a course must be able to edit courses.	The system should provide the course owner the option to edit the course's details such as name, department and institution. The system should display a text input form to the owner, meant for adding the new course's details.
FR12	The owner of a course must be able to add announcements.	The system should provide the course owner with a button of adding a new announcement to the course. The system should display a text input form to the owner, meant for adding the announcement's text.
FR13	The owner of a course must be able to add files.	The system should provide the course owner with a button of uploading a file to the course. The system should display a file manager dialog to the owner, meant for choosing the file which will be uploaded to the course.

FR14	The system must provide the course announcements in a list.	The system should provide all the members of a course with an announcement button that redirects them to a list with all the course announcements.
FR15	The system must provide the course files in a list.	The system should provide all the members of a course with a file button that redirects them to a list with all the course files. Each file should be downloadable.
FR16	The owner of a course must be able to create a new poll.	The system should provide the course owner with a new poll button that redirects the user to a panel with all the options of creating the poll form.
FR17	The owner of a course must be able to delete a poll.	The system should provide the course owner with the option of deleting a poll. After deletion, the poll will not be available to course members.
FR18	The owner of a course must be able to lock/unlock a poll.	The system should provide the course owner with the option of locking/unlocking a poll. After locking, members of the course will not be able to open the poll, but they will be able to view the poll's results. After unlocking, members of the course will be able to open the poll.
FR19	The system must provide the course's polls in a list.	The system should provide all the members of a course with a poll button that redirects them to a list with all the course's polls.
FR20	Course members must be able to view/vote a poll.	The system should allow course members to view and vote in the available polls.
FR21	Course members must be able to view the results of a poll.	The system should allow course members to view the results of the available polls. For example, for each question, the system should display the statistics of the members' answers.
FR22	Course members must not be able to open a poll if they have already voted.	The system should not allow course members to open the poll after the voting action in the poll.
FR23	The owner of a course must be able to delete the announcements.	The system should allow the owner of a course to delete the announcements. After deletion, the deleted announcement will not be available to the course members.
FR24	The owner of a course must be able to edit the announcements.	The system should allow the owner of a course to edit an announcement. After edition, the system will update the announcement's details.
FR25	The owner of a course must be able to delete the files.	The system should allow the owner of a course to delete the files. After deletion, the deleted file will not be available to the course members.
FR26	The system should provide recognition messages for any action that is available in each course.	The system should provide messages for any user's actions in a course. For example, after adding or deleting an announcement, a file or a poll, the system will show a message of the action's completion. For any error in an operation, the system will display the error details in a message.

FR27	The system should check if user has been added to a new course.	The system should provide a notification to user, when he has been added to a new course. Pressing the notification will redirect user to a news feed window.

Table 4.1: Functional Requirements

4.2.2 Non-Functional Requirements

A non-functional requirement describes not what the system will do, but how the system will do it, for example, system performance requirements, system external interface requirements, design constraints, and software quality attributes. Generally, non-functional requirements are expressed in the form “system shall be <requirement>”. [4]

Below, the most important classes of the non-functional requirements are presented based on their importance: [46]

- **Usability**: is the ease with which a user can learn to operate, prepare inputs for, and interpret output of system.
- **Modifiability/Extensibility**: The ability of the system to be easily changed to meet new requirements.
- **Portability**: The ease with which a system or component can be transferred from one environment to another.
- **Flexibility**: The ability of the system to easily exchange information with the user.
- **Reliability:** is the ability of a system to perform its required functions under stated conditions for a specific period of time.
- **Security:** login, password requirements.
- **Performance**: concerns the speed of operation of a system.
- **Total Cost**: The total cost of the project in terms of price and time is described by this quality attribute.

The system’s related non-functional requirements are given below:

ID	Description	Explanation
NFR01	The graphical user interface must be easy enough for anyone and all screens should have a similar style.	The buttons, menus and layouts should be the same in all the screens of the application. The users will execute specific actions in a certain way. They will find the same options and menus in each screen, while pressing the navigation buttons.
NFR02	The system should show clear and detailed notification messages to the user.	The system should display messages in pop-up windows with details regarding the status of any operation. If something needs the attention of the user, the system will display the notification message for it. For example, the completion of adding a new course or adding a new member.
NFR03	The system must have lack of bugs and inform the user of every wrong operation.	The system must be checked for any possible bugs in its operations before it is released to users. Also, it must provide log error messages (notification messages) to users to inform them of any wrong operation.

NFR04	The system will be able to run on all Android devices.	The system will be developed to run on all Android devices, such as mobiles, tablets or any other device that uses the Android operating system.
NFR05	The system will request a password for each user account.	The system will not display the available features to a user unless he logs in to his account.
NFR06	The system will have fast response time.	The system should provide all its operations very fast. The user must not wait for any operation for a long time. If something needs time to be executed, then a spinning loader will be displayed until the end of the operation.
NFR07	The system must be designed to be able to accept new operations and features.	The system must be designed in such a way that any developer can add new operations and features to the source code.

Table 4.2: Non-Functional Requirements

4.3 Use Cases

In software and systems engineering, a use case is a list of steps which defines interactions between actors and a system, to achieve a goal. An actor is a person, organization or external system that plays a role in one or more interactions with the system. [47]

4.3.1 Use Case Diagram

In our project, the actors are the primary users from the stakeholders' analysis. Figure 4.1 presents the use case diagram of the system. As the diagram shows, there are two actors, course participants and course owners, with the last to inherit the interactions of the first.

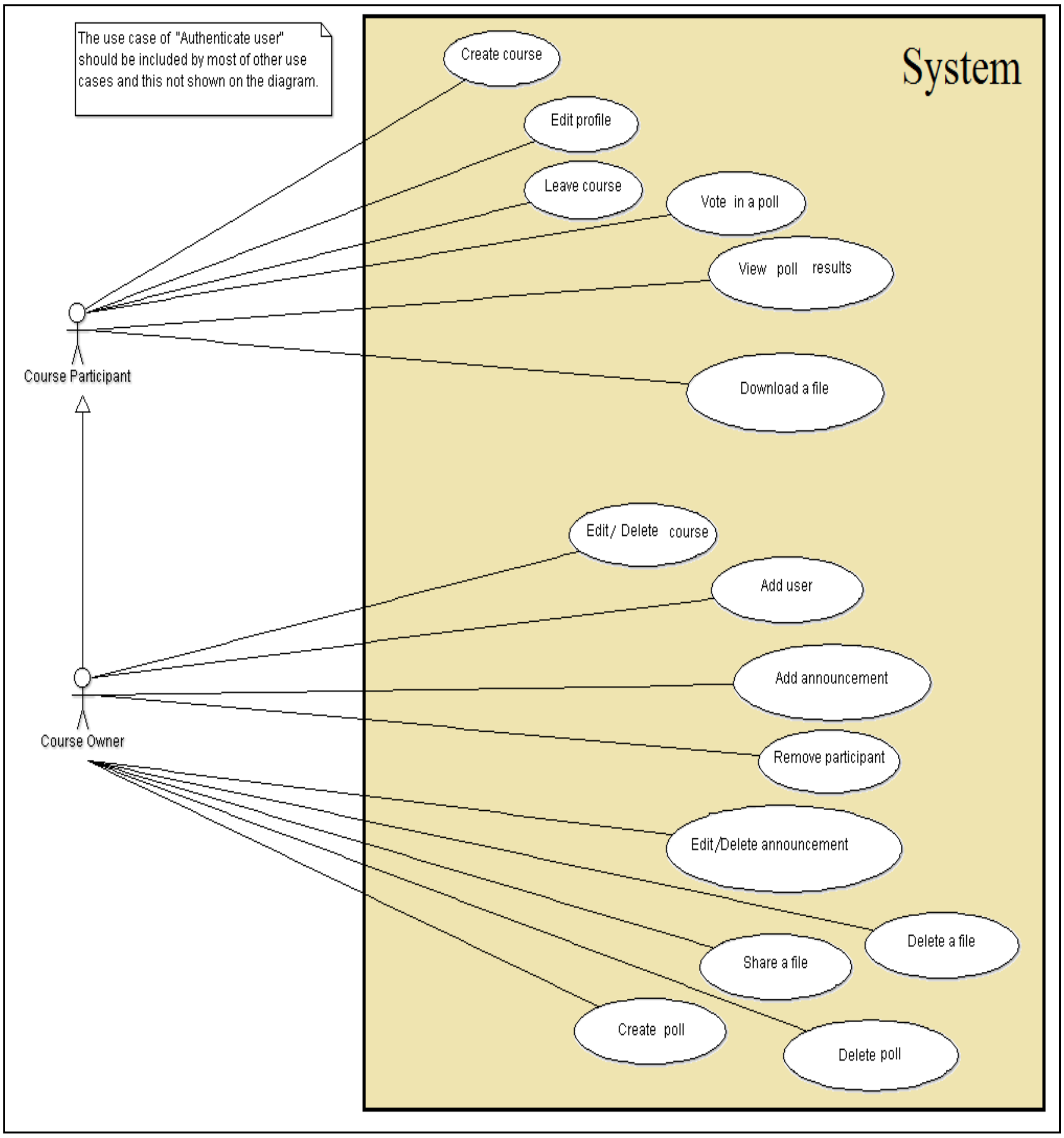


Figure 4.1: Use Case Diagram

4.3.2 Use Case Scenarios

In this section, only two use case scenarios are provided, one for each use case actor. The rest of use case scenarios can be found in Appendix A. The first use case scenario in Table 4.3 describes the way a course owner can create a new poll in the course. The second scenario in Table 4.4 describes the way a course participant can vote in a course's poll.

Use Case Name	Create poll
Context	Create a new poll
Primary Actor	Course Owner
Precondition	Application is running, user is logged on and user selected a course
Success Post Condition	New poll was added correctly
Trigger	User wants to add a new poll
Main Success Scenario	**1.** User presses on polls **2.** User presses on new button **3.** System displays a window with all options to create a poll form **4.** User creates the poll fields and choices and presses confirm **5.** System closes dialog if the poll has been added and shows message about the action **a.** If poll name exists a message is displayed to user Exception **b.** If there is a problem with the internet connection an error message is displayed to the user **Exception**

Table 4.3: Create poll use case scenario

Use Case Name	Vote in a poll
Context	Vote in a poll
Primary Actor	Course Participant
Precondition	Application is running, user is logged on, user selected a course, user pressed on polls
Success Post Condition	User has voted
Trigger	User wants to vote in a poll
Main Success Scenario	**1.** User opens a poll **2.** System displays the poll form **3.** User chooses his answers and presses confirm **4.** System closes the form if user has voted correctly and shows message about the action **a.** If there is a problem with the internet connection or with the voting operation an error message is displayed to the user **Exception**

Table 4.4: Vote in a poll

Chapter 5: Design

This chapter presents the design of the mobile application and its web server, whose purpose is to meet the requirements defined in Chapter 4. The design chapter starts by addressing the software's architectural design followed by the software's design model, the web server design and the user interface design. As for the design phase, the software's model, web server and user interface were done together. The order in which the sections are presented doesn't have anything to do with the order of the design process, since the agile model was followed (Section 1.3.5).

5.1 Architecture Design

This project requires the establishment of communication between the mobile application and a web server, exchanging messages to support a wider communication between all the mobile phones that are using our application. We used the client-server architecture to establish the communication process, which is similar to that of the HTTP protocol [48], used by web browsers.

The client-server architecture is an architecture in which many remote processors (clients) request and receive service from a host computer (server). The services running on the server run on known ports which are the application identifiers, and the client needs to know the address of the server machine and the port to connect to the server. Although the client must know the server's address and port, the server does not need to know anything about the client's address or port at the time of connection initiation. The client acts as the active device which makes the first move to establish the communication whereas the server passively waits for such a request from a client.

To better understand the entire project architecture, Figure 5.1 shows the component view of the system. Each component of the figure will be analysed in the next sections. Although a client-server architecture is used, the server is used only for communication and storage. Users do not need extra information, as they can utilize all the application's functions from their mobile phone.

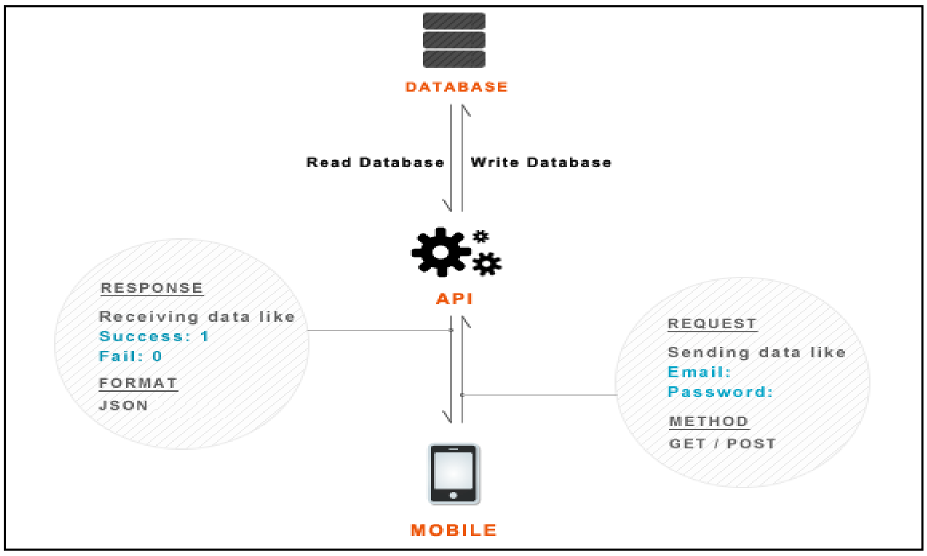


Figure 5.1: Component View of the System

The application uses a parser to pass requests to the web server and to receive responses from it using HTTP GET/POST [48] requests and responses. Each request contains information about which method is to be invoked and all the arguments for the method to call. The server's response contains the result of a service method invocation. The server application uses a connection to communicate with a database which is used to store and load data. The web server consists of PHP files that contain methods for the database processes and the communication between the server and the application.

5.1.1 Technologies Used

PHP is a server-side based language which means that PHP code can be executed on computers with a PHP processor module. The latest version of PHP is PHP 5.5.2 [49]. In order to run the PHP code, an apache server must be run on the web server. After the installation of the apache, the server is able to accept HTTP requests from any mobile device. MySQL Server [50] is also installed on the web server to provide a database for storing the data into tables (Section 5.3.2).

Figure 5.1 shows the web server's response format. JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Based on JavaScript object literals, JSON is a textual representation of complex data instances. It has four basic types (Strings, Numbers, Boolean, Null). Out of these four types objects or arrays can be constructed where the object in terms of JSON is used for key-value pairs. [51]

5.2 Design Model

This section describes in detail the structure of the system and how the system will be implemented. It represents the application components and determines their appropriate placement and use within the overall architecture. The section starts presenting the packages that contain the design elements of the system, followed by the system's classes.

5.2.1 Package Diagram

Android software development doesn't follow a structural design, or pattern [52] and it is difficult for other developers to maintain the existing code and work on it. Thus, we decided to organize our application's class files into packages, each of which is to contain classes that belong under the same operational category.

The system contains three main packages (activity, manager, service). The activity package contains two sub packages (account, course) and the course sub package contains one sub package (poll) (Figure 5.2).

activity: activity contains all classes that represent the application's activities (screens) and their operation. It is responsible for starting the application and displaying the screens on the mobile device. It uses the service package to establish the communication between the application's activities and the web server. It also uses the manager package to manage its own data, that are being produced while the application is running and runs operations that check the connection availability.

manager: manager contains four classes (AlertDialogManager, ConnectionManager, CourseSessionManager, and SessionManager). This package is responsible for keeping temp data that are used by the application and for checking for network connectivity changes.

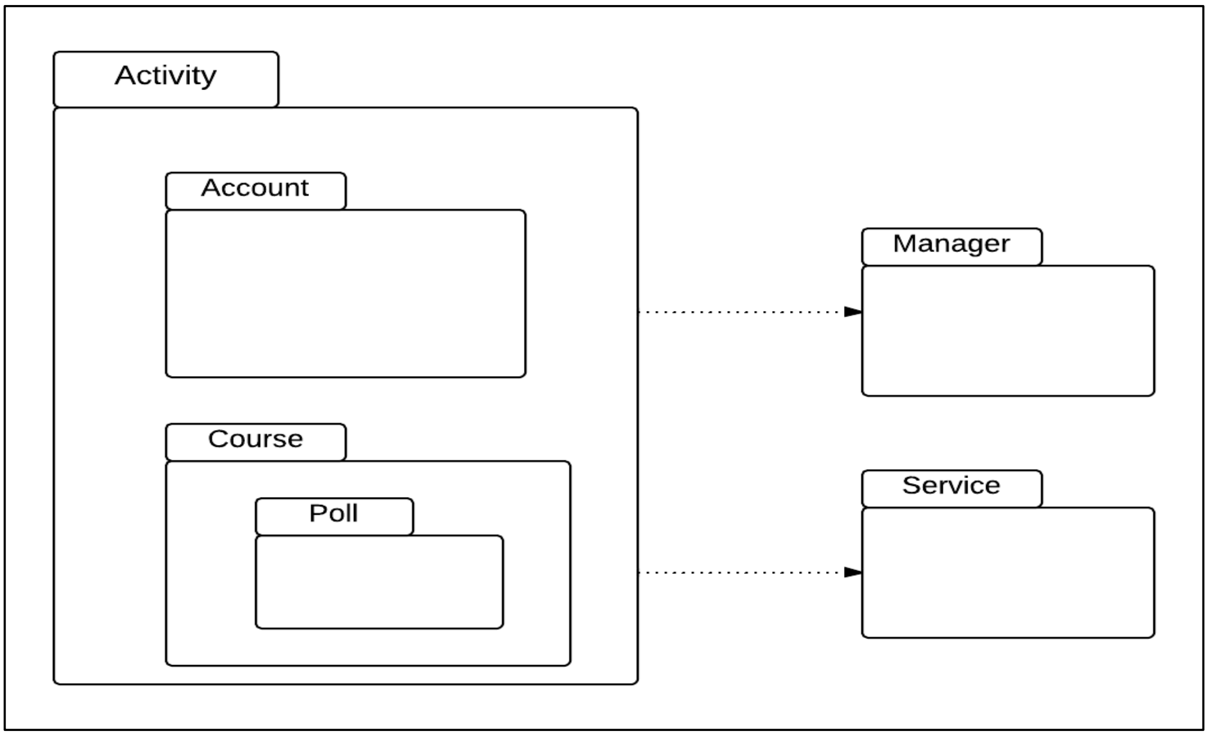


Figure 5.2: System's Package Diagram

service: service contains the BackgroundService, JSONParser and ServiceFunctions classes. These classes provide the communication between the application and the web server, creating the requests and parsing the web server's responses.

account: account is the sub package of the activity and contains the classes that are responsible for sign in, register, change password, and change account name operations and activities.

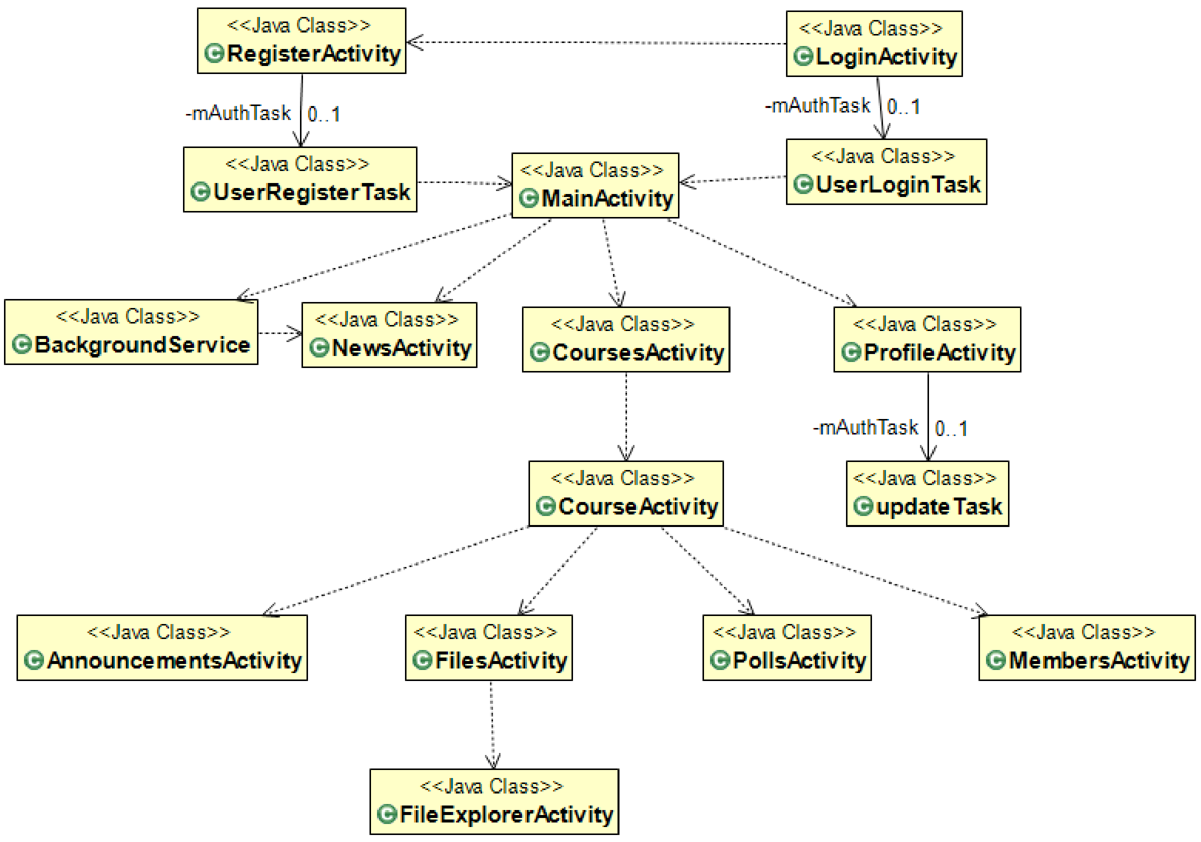
course: course is the sub package of the activity and contains the classes that are responsible for the display of the course's activities and provides their functionality.

poll: poll is the sub package of the course and provides the classes that display the poll activities as well as their functionality.

5.2.2 Class Diagram

The following simplified diagram presents the main classes of the system. It is a reference to the extended class diagram which includes the attributes and operations of the classes. The extended class diagram is available in Appendix B.

Also, in the diagram below, the classes that belong to the manager package and the classes (JSONParser, ServiceFunctions) are not presented, because each one of the other classes have dependencies (arrows) on them and thus the diagram is not easily readable. An extended diagram with the manager package, the JSONParser and the ServiceFunctions classes is also available in Appendix B.



<<Java Class>> RegisterActivity
<<Java Class>> LoginActivity
-mAuthTask 0..1
-mAuthTask 0..1
<<Java Class>> UserRegisterTask
<<Java Class>> MainActivity
<<Java Class>> UserLoginTask
<<Java Class>> BackgroundService
<<Java Class>> NewsActivity
<<Java Class>> CoursesActivity
<<Java Class>> ProfileActivity
-mAuthTask 0..1
<<Java Class>> CourseActivity
<<Java Class>> updateTask
<<Java Class>> AnnouncementsActivity
<<Java Class>> FilesActivity
<<Java Class>> PollsActivity
<<Java Class>> MembersActivity
<<Java Class>> FileExplorerActivity

Figure 5.3: Simplified Class Diagram

Each class which has the suffix Activity represents a screen of the mobile application. For example, LoginActivity.class and RegisterActivity.class are the login and register screens respectively. The classes that contain the word task, such as UserRegisterTask.class, UserUpdateTask.class and UserLoginTask.class, are asynchronous task classes that are running for a short period of time in the background to support the Login, Register and Update user's data operations.

BackgroundService.class is a class which is running from the moment that MainActivity.class starts running. It is running during the application's entire life time in the background and it checks for any changes to the user's data in the web server's database and displays a notification that triggers the running of NewsActivity.class.

The diagram shows the relationships (dependencies) between the classes. When an activity class depends on another activity class in our design, it means that the independent class is called by the depended class. In other words, if we want to call (display) MainActivity.class (main screen), we must use LoginActivity.class or RegisterActivity.class.

As mentioned above, the classes of the manager package and JSONParser, as well as ServiceFunctions classes are not displayed on Figure 5.3, but we will give a description for them. The classes that exist on diagram Figure 5.3 depend on all the manager package classes. Each class needs to know if it can have access to the internet and, for that reason, it uses the manager package. They also need to have access to data (user name, password, email) that are initially stored in the application from the application start up. Also, all classes on Figure 5.3 depend on ServiceFunctions.class, which is responsible for the communication between the application and the web server. Each class needs to use the methods from the ServiceFunctions.class to receive or send data to the web server. ServiceFunctions.class depends on JSONParser.class, which provides the parser functionality before the HTTP GET/POST request and response.

5.2.3 Sequence Diagram

A sequence diagram is used to show the interactions between objects in the sequential order that those interactions occur [53]. Thus, with a sequence diagram it is easier to understand the data and messages flow between the activity, manager, and service package objects. Figure 5.4 shows the automatically generated sequence diagram for the retrieval of the user's course list**,** and Figure 5.5 shows the automatically generated sequence diagram illustrating the process of the user submitting a vote in a poll. The processes depicted in these figures may not be clearly visible, but we are including them in the study simply to show that they are available**.** The rest of the sequence diagrams can be found in Appendix B.

When a user is navigating through the Courses Screen, the activity tries to access the user's courses from ServiceFunctions.class, which uses a JSONObject to communicate with the server. If there is no error, it checks if the web server response has a successful message (operation done correctly) and it extracts the data for each course (name, institution, department, owner_id) in a loop, and adds them to the defined lists (courseDepartmetList, courseOwnerList, courseInsitutionList, courseNameList).

Figure 5.5 shows that when a user presses to submit his vote, the PollVoteActivity checks if he has selected an option from the voting list (selected == -1). If the user has selected an option, it checks for an available internet connection. Then, it tries to check if the poll is still open using the ServiceFunctions.checkPollStatus() (maybe the owner of the poll has closed it at the time that the user pressed to submit the vote) and, if it is still open, it submits the vote using ServiceFunctions.votePoll().

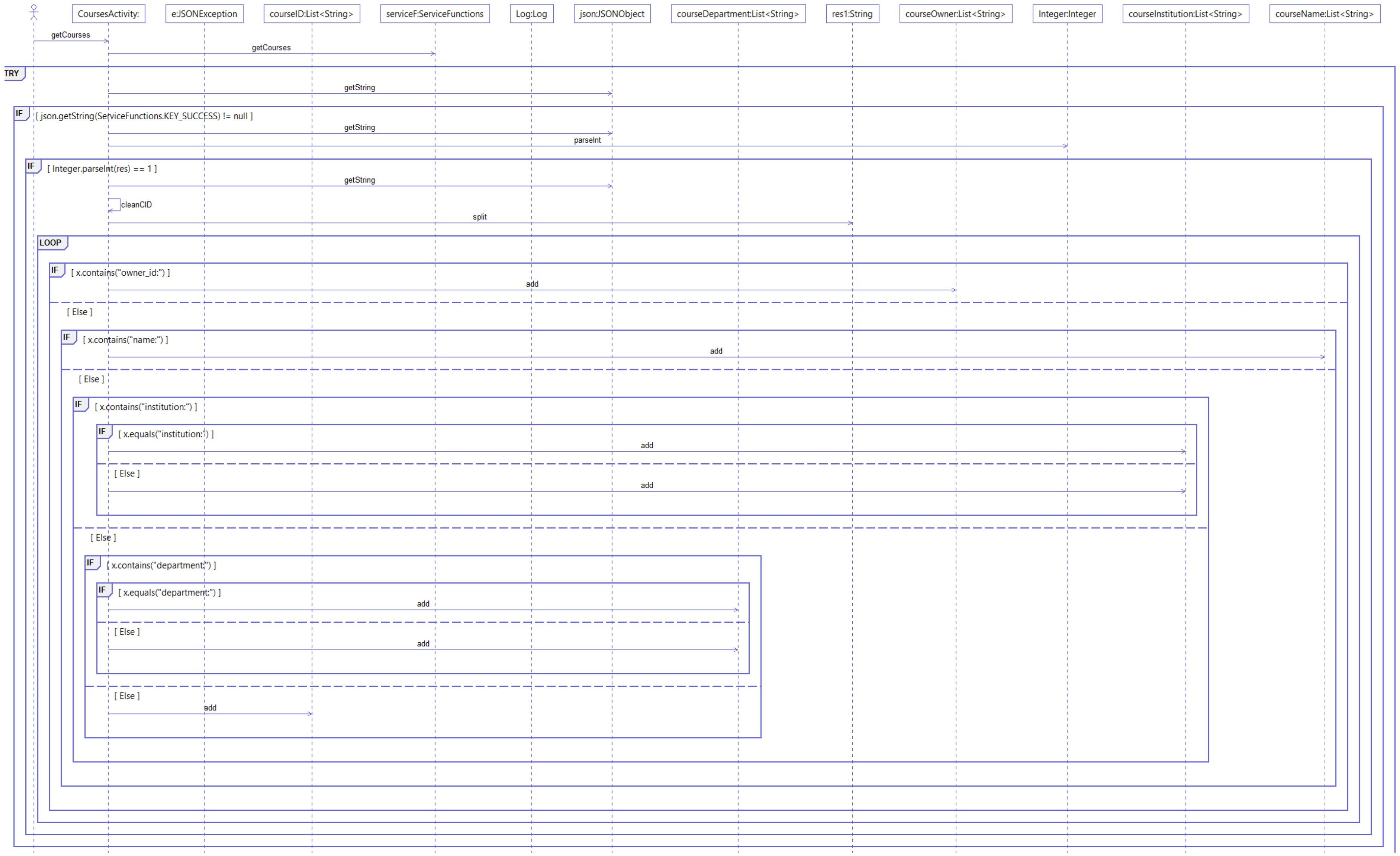


Figure 5.4: Retrieve User's Courses List Sequence Diagram

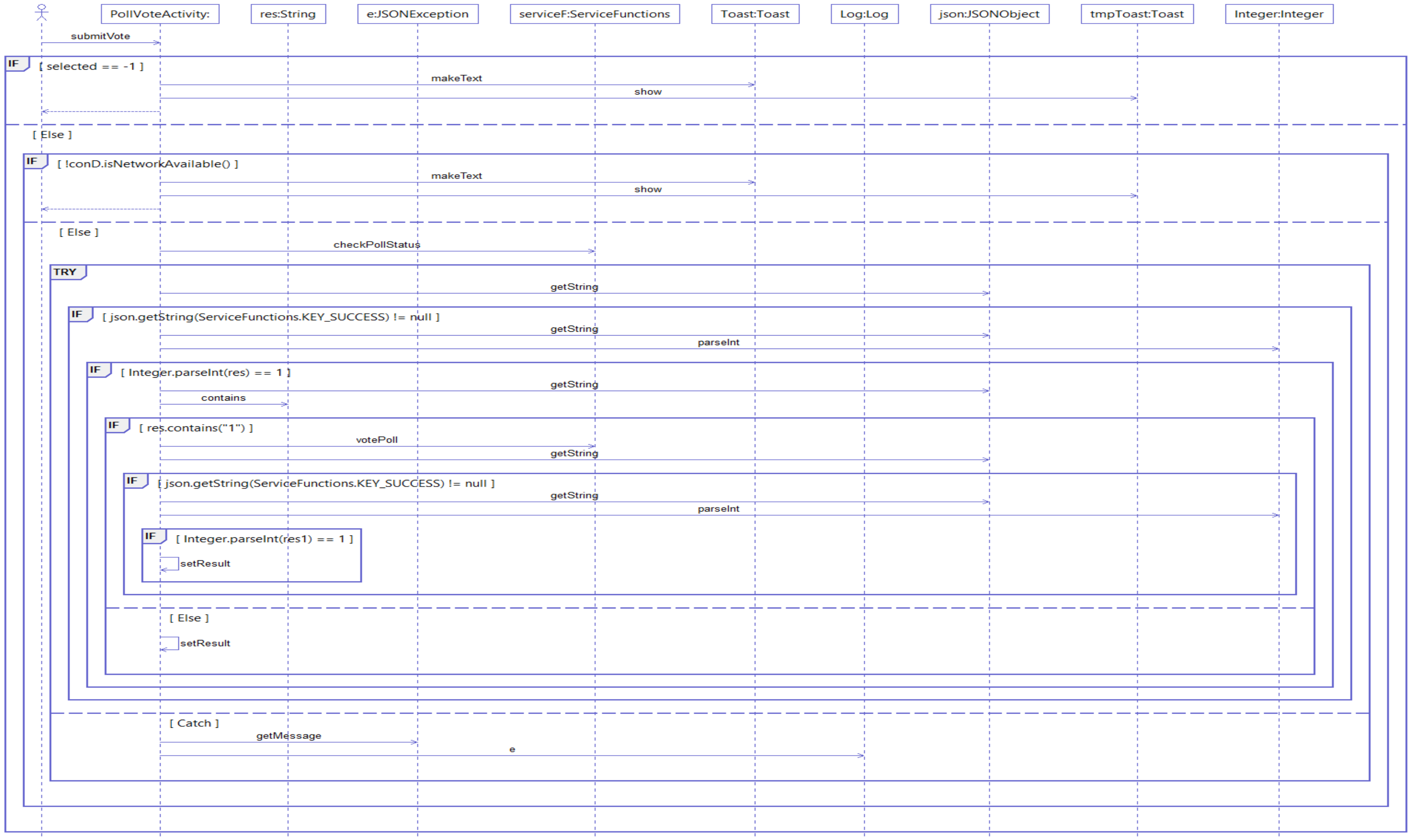


Figure 5.5: Submit Vote in a Poll Sequence Diagram

5.3 Web Server Design

This section presents the design of the web server and how it will be implemented. It starts with the structure of the web server, followed by the database structure.

5.3.1 Web Server Structure

Figure 5.6 shows the web server structure. It consists of four PHP files (index, DB_Functions, DB_Connect, uploadFile). A description of each web server component is given below:

index: the index file is responsible for handling all requests. It accepts HTTP GET/POST methods to support the client-server communication. Each request is to be identified by a TAG, which is a non-hierarchical keyword assigned to a piece of information. Depending on the TAG, index will call a specific method from the DB_Functions file. If TAG is not defined in the index file, then an access denied message will be sent to the requestor, otherwise the response will be in a JSON format [51].

Web Server
php
index
php
DB_Functions
php
DB_Connect
php
uploadFile

Figure 5.6: Web Server Structure

DB_Functions: A class which contains methods to insert/read/update operations on the database. It requires the DB_Connect file to establish the connection to the database.

DB_Connect: A class which contains methods to connect or disconnect from the database.

uploadFile: It is responsible for accepting files from the application and moving them to a specific directory.

5.3.2 Database Structure

Figure 5.7 below shows the entity relation diagram of the web server's database. The table users is used to store the users. The table courses is used to store all the courses that have been created by the users. The table users_news is used to store all the users' news. The courses_polls, courses_announcements and courses_files tables are used for keeping the announcements data, files data, and polls data of each course. The users_courses table is used to match users to courses. Similarly, users_polls table is used to match users to polls.

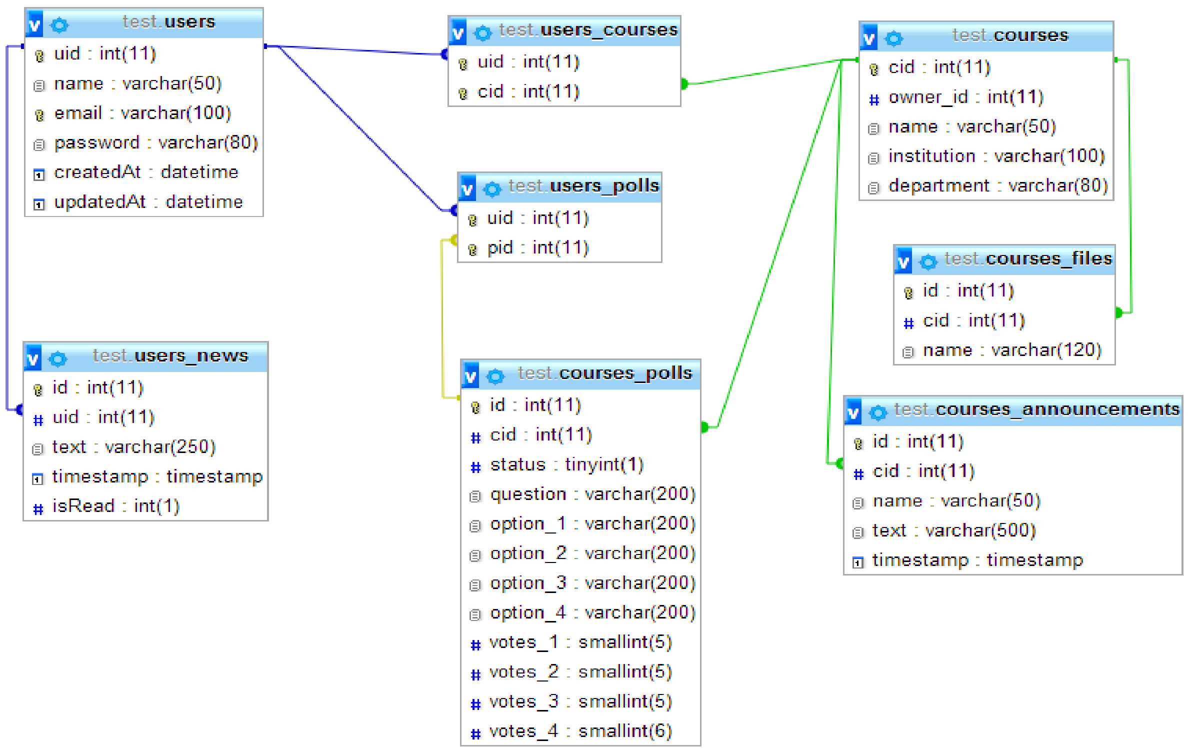


Figure 5.7: The entity-relation diagram of webserver's database

In all tables, except users, users_courses and users_polls, there is a primary key, which means that it should have a unique value. The tables users, users_courses and users_polls have two primary keys, which means that they do not accept a row with the same primary keys. The lines that connect the tables represent a foreign key relation, which means that the foreign key of one table points to a primary key in another table. Foreign keys are used in our database design to prevent invalid data from being inserted and also to accelerate the deletion operations in the database. Thus, when a primary key is deleted, all the rows that have a foreign key which points to it are deleted too, without the need of an extra database command. All the attributes of tables users and courses are explained in Table 5.1 and all other tables are explained in Appendix B.

users		
Attribute	**Type**	**Description**
uid	INT	The unique id of the user.
name	VARCHAR	The name of the user.
email	VARCHAR	The unique email of the user.
password	VARCHAR	The password of the user.
createdAt	DATETIME	The time that the user is created.
updatedAt	DATETIME	The time that the user's details are updated.
courses		
Attribute	**Type**	**Description**
cid	INT	The unique id of the course.
owner_id	INT	The owner id of the course.
name	VARCHAR	The name of the course.
institution	VARCHAR	The name of the course's institution.
department	VARCHAR	The name of the course's department.

Table 5.1: Database Tables' Attributes

5.4 User Interface Design

In order for the application to satisfy the requirements and maximize the usability, much emphasis of the design is put on the user interface. The principles and guidelines that are mentioned in Chapter 3 can be identified in the sections below.

5.4.1 Layout Structure

As mentioned in Chapter 3, the application's screens must follow some principles. Thus, three layouts are used for the application's interface, to support all orientations and provide scrolling for the devices with small screen. Each application screen is using either LinearLayout or RelativeLayout wrapped by ScrollView to provide the above feature. Figure 5.8 shows the register screen in both orientations and the scrolling availability in landscape orientation.

5.4.2 Theme and Icons

The same theme is used for all screens. The combination of the (blue, white, light grey) colours provides a friendly environment to the user. Buttons have either a blue or a grey colour, and their colour changes when clicked, to inform the user that the action has been taken. Icons are used for the main screen window and for the menu control buttons to represent the subtle visual and spatial concept. Figure 5.9 shows the theme of the application and the icons that have been used in the Main and the Profile screens. The Main screen can redirect the user to the Courses, Profile and News Feed screens. The Profile screen shows the name and email of the user. The user can change his name and password.

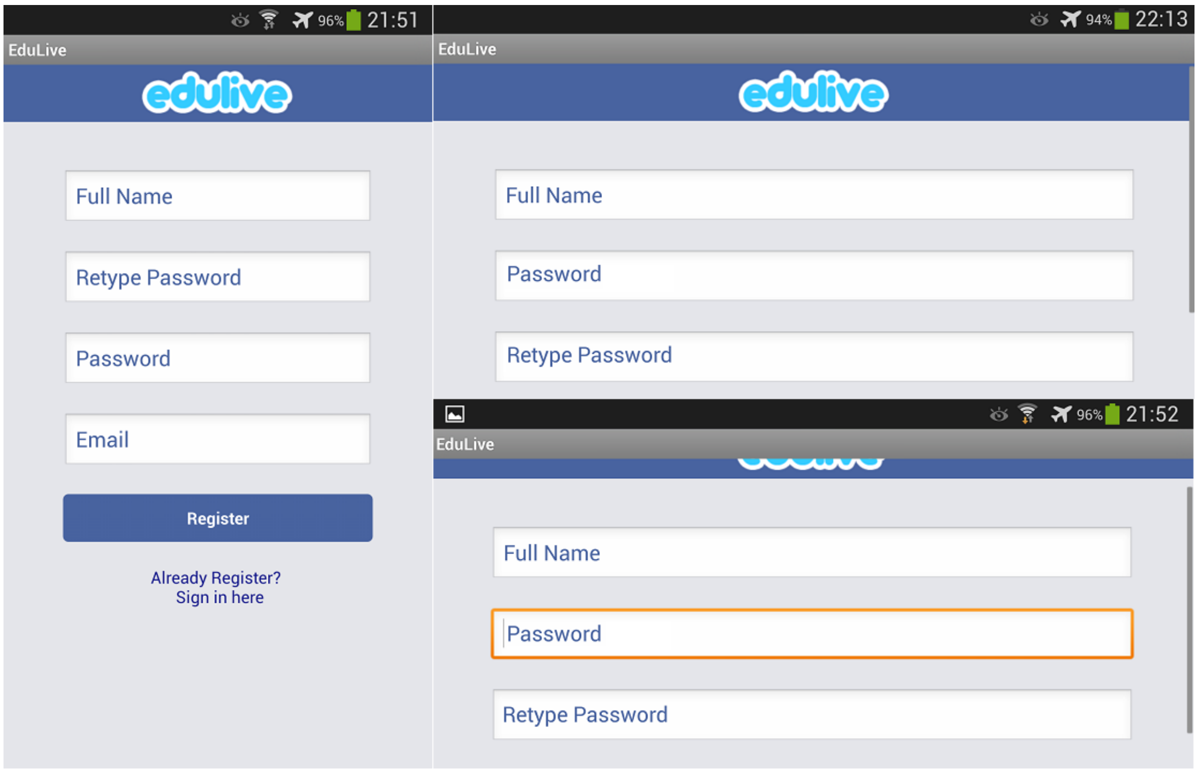
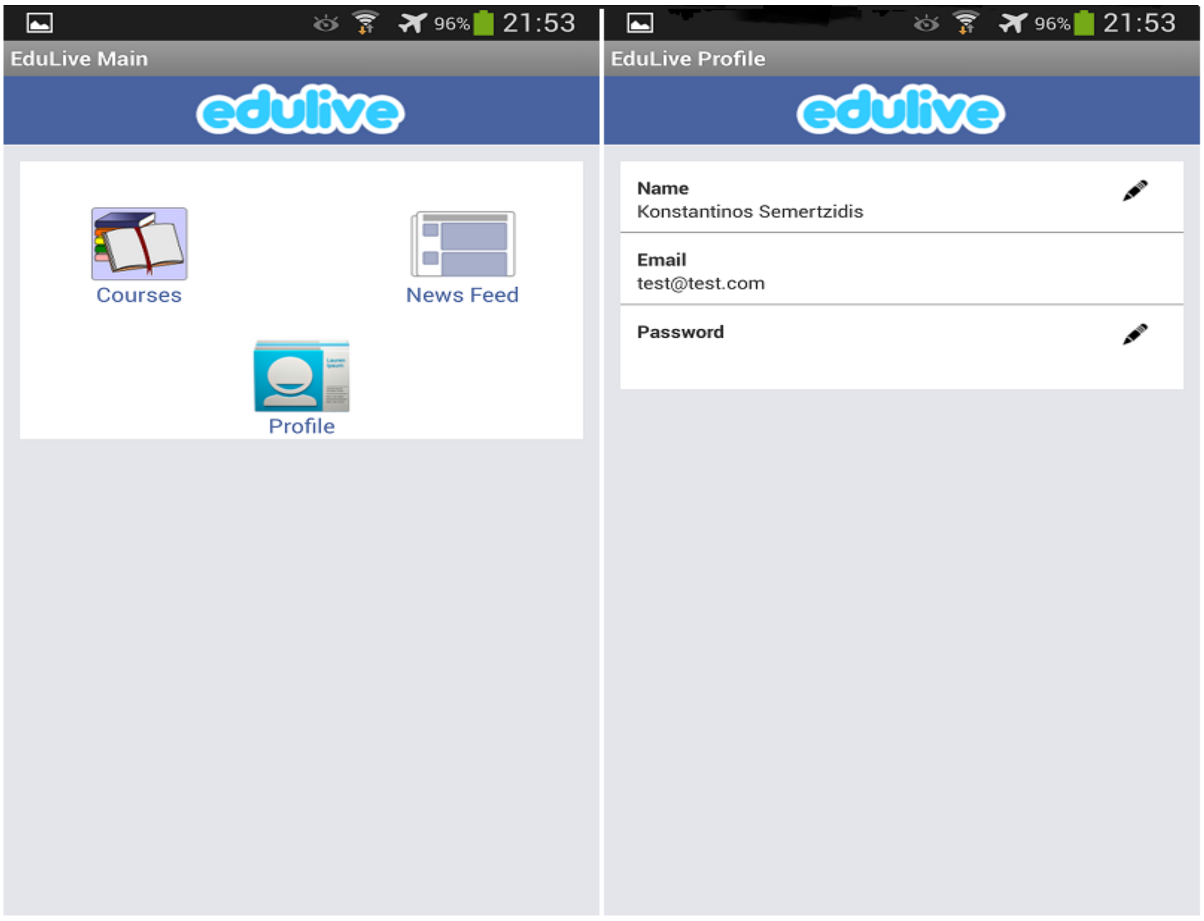


Figure 5.8: Register Screen in all orientations with scrolling in landscape



Name
Konstantinos Semertzidis

Email
test@test.com

Password

Figure 5.9: Application's Theme and Icons

5.4.3 Navigation and Controls

Each screen has navigation to other screens. The Login, Register and Course screens (activities) use buttons to navigate to the main screen. Main screen uses springboard (Figure 5.9) for navigation to the other screens. Courses, Polls, Announcements, Files and Members screens use lists to transfer user to another screen. All screens that use list navigation provide a search feature (mobile design pattern Section 3.2) to scan the data faster.

As mentioned in Section 3.1, controls should be beneath content, thus in the application all controls are displayed near the device control buttons. Figure 5.10 shows an example of the application's controls and navigation in Courses and the Course screen. From the Course screen the user can navigate to the Polls Announcements, Files and Members (if the user is the owner of the course) screens. Controls can provide a user with the features of logout, redirect to home screen (Main) and refresh the data in a list, or create new data such as a new course.

5.4.4 Feedback and Alert Dialogs

Feedback and alert dialogs exist in the application. Alert dialog is used to show the progress of an operation, to avoid user's frustration for any delay of the application or to request the user's input for an action. We took a step further in the display of feedback messages and instead of displaying alert dialogs that hide the content of the application we used toasts. Toasts can provide simple feedback on an operation in a small popup, they only fill the amount of space required for the message and the ongoing activity remains visible and interactive.

Figure 5.11 shows the alert dialogs and feedback messages in the Course's File screen, which is a screen for downloading/uploading files to the course. Thus, it shows an alert dialog which requests a user's input for a file deletion, an alert dialog which shows the progress of a file's download, and a toast message which shows that a file has been uploaded correctly.

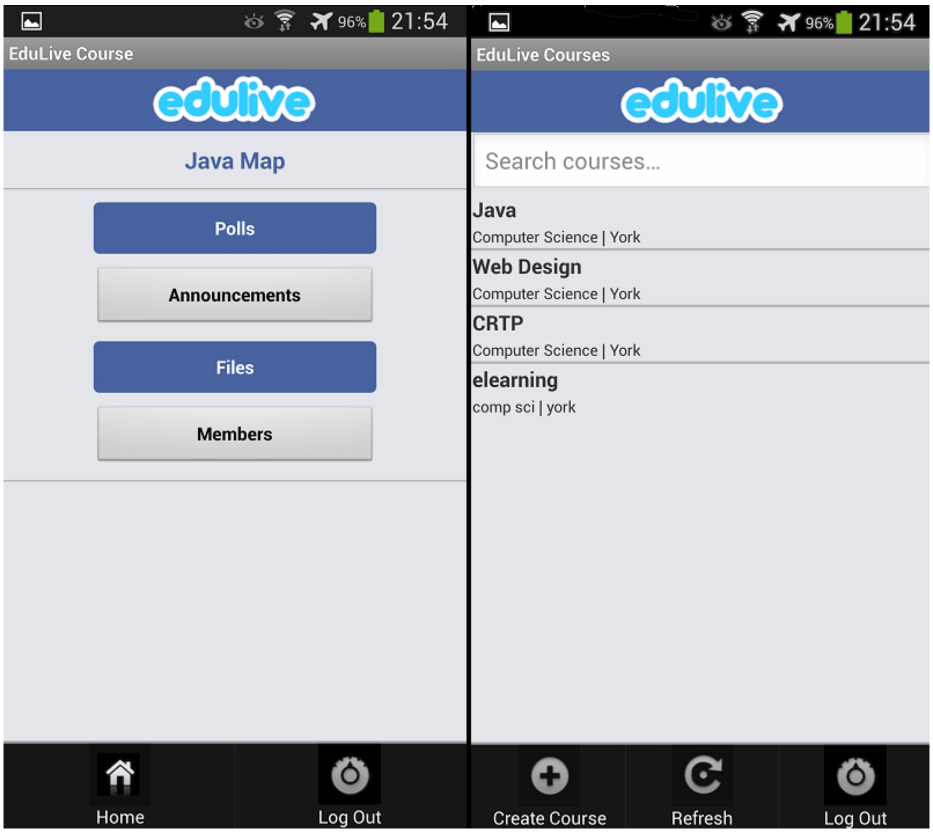
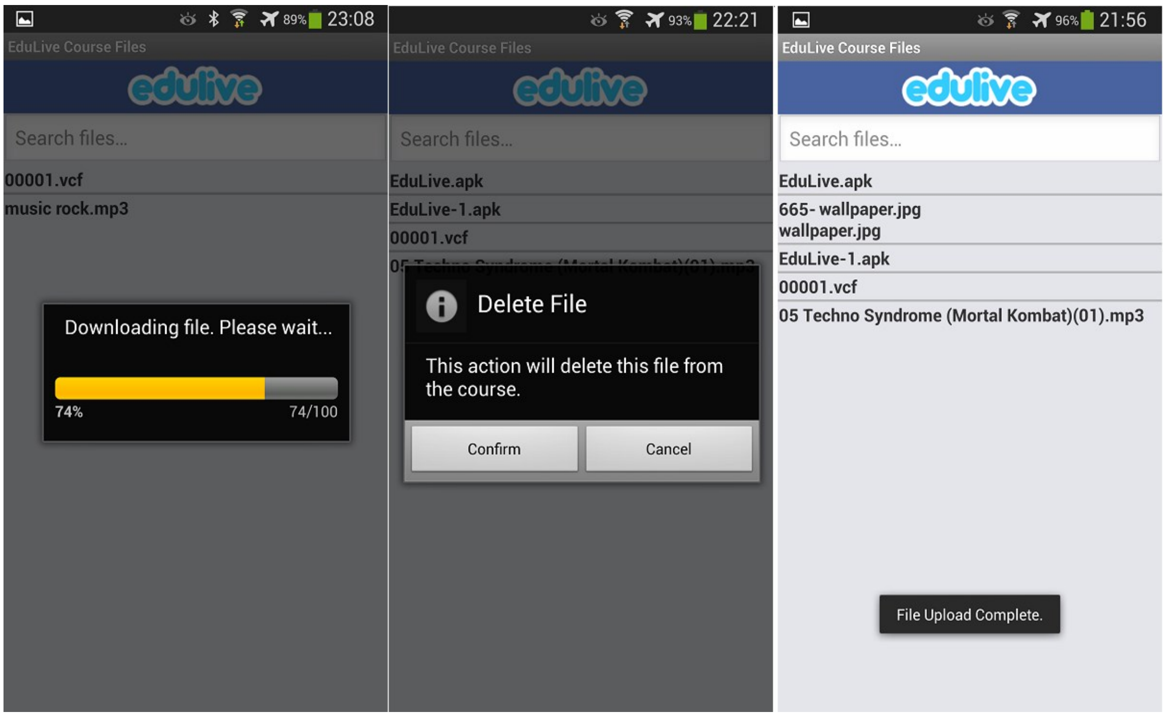


Figure 5.10: Application's Navigation and Controls

Also, Figure 5.12 shows the alert dialog forms, presenting the forms of creating a new course and an update password. Thus, it shows an alert dialog which request a user's input for a new course creation, and an alert dialog which request a user's input for the user's password update. All the other alert dialog forms which are used in the application are given in Appendix B.



EduLive Course Files

00001.vcf
music rock.mp3

Downloading file. Please wait...
74% 74/100

EduLive Course Files

EduLive.apk
EduLive-1.apk
00001.vcf

Delete File

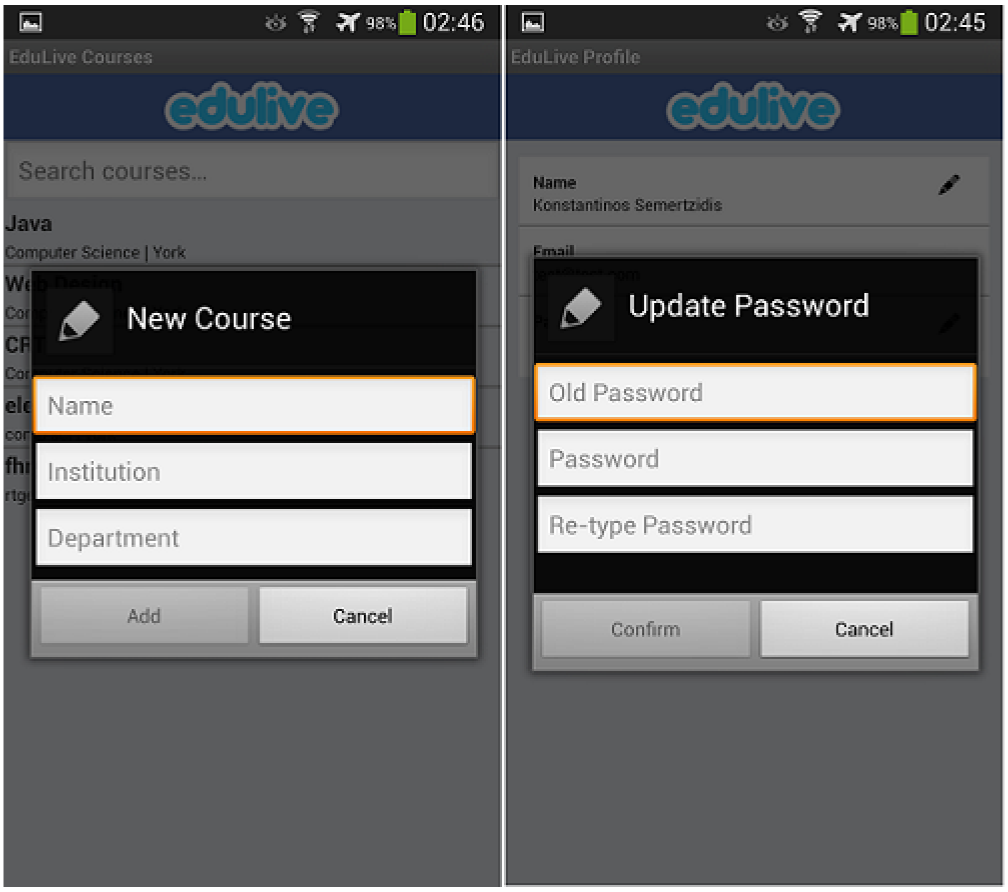
This action will delete this file from the course.

EduLive Course Files

EduLive.apk
665- wallpaper.jpg
wallpaper.jpg
EduLive-1.apk
00001.vcf
05 Techno Syndrome (Mortal Kombat)(01).mp3

File Upload Complete.

Figure 5.11: Applications Alert Dialogs and Feedback



EduLive Courses

Java
Computer Science | York

New Course

EduLive Profile

Name
Konstantinos Semertzidis

Email

Update Password

Figure 5.12: New Course and Update Password Alert Dialog Form

Chapter 6: Implementation

This chapter presents the implementation of the application and the web server design, presented in the previous chapter. The first section presents the application development followed by the section of the web server development. In each section the tools that have been used are also presented.

6.1 Application Development

This section presents the implementation of the mobile Android application. Based on the design described in Chapter 5, the code written and the implementation techniques used are presented.

6.1.1 Tools - Libraries Used

The implementation of the Android application was done entirely in the Eclipse IDE using the ADT plug-in and Android SDK. All the tools that are needed to design, create and implement the application are included in that platform (Section 2.4). In order to implement the feature of showing the voting results in a chart (PollResultActivity.class), we used the AChartEngine library. AChartEngine is a charting library for Android applications which supports a variety of chart types [54].

6.1.2 Managers Implementation

The implementation of the manager package is needed to keep data running during the application and to check and provide information about the connection.

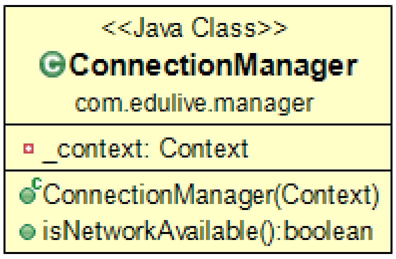


Figure 6.1: ConnectionManager Class Diagram

```
public class ConnectionManager {
    private Context _context;

    public ConnectionManager(Context context){ this._context = context; }
    public boolean isNetworkAvailable()   {
         ConnectivityManager connectivity = (ConnectivityManager)
        _context.getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo activeNetworkInfo = connectivity.getActiveNetworkInfo();

           return activeNetworkInfo != null && activeNetworkInfo.isConnected(); }}
```

Listing 6.1: Connection Manager

Figure 6.1 shows the class diagram of ConnectionManager.class and Listing 6.1 shows the implementation of ConnectionManager.class. It takes as an argument the Context of the current state of the object which created the connection manager object. ConnectionManager.class contains the isNetworkAvailable() method, which returns if there is any internet provider available. Thus, if an activity needs to check if internet is available, it can create a ConnectionManager object and call the isNetworkAvailable() method.

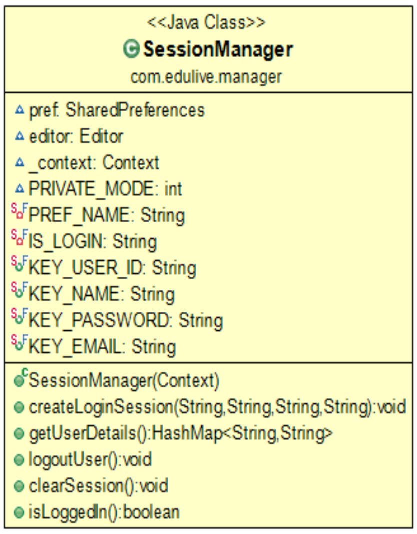


Figure 6.2: SessionManager Class Diagram

In the manager package, Session and Course Manager have also been implemented. Listing 6.2 shows an extract of the SessionManager.class code and Figure 6.2 the SessionManager class diagram.

SessionManager.class uses SharedPreferences, which is used in Android to simply store and retrieve fixed amounts of data. SessionManager is used by activities to get the user's details (name, email, and password), which have been entered during the login process. Thus, in Listing 6.2 the createLoginSession() method is to store the user's data and the getUserDetails() method is to get the data. The CourseSessionManager.class implementation is similar to SessionManager's, the only difference being that it stores the details of a course (id, name, owner_id).

```
public class SessionManager {
    SharedPreferences pref;
    Editor editor;
    Context _context;
    private static final String PREF_NAME = "EduLivePref";
    private static final String IS_LOGIN = "IsLoggedIn";
    public static final String KEY_USER_ID = "uid";
    public static final String KEY_NAME = "name";
    public static final String KEY_PASSWORD = "password";
    public static final String KEY_EMAIL = "email";

    public SessionManager(Context context) {
        this._context = context;
        pref = _context.getSharedPreferences(PREF_NAME, 0);
        editor = pref.edit();
}

    public void createLoginSession(String userID, String name, String email, String password){
        editor.putBoolean(IS_LOGIN, true);
        editor.putString(KEY_USER_ID, userID);
        editor.putString(KEY_NAME, name);
        editor.putString(KEY_EMAIL, email);
        editor.putString(KEY_PASSWORD, password);
        editor.commit();
}

    public HashMap<String, String> getUserDetails() {
        HashMap<String, String> user = new HashMap<String, String>();
        user.put(KEY_USER_ID, pref.getString(KEY_USER_ID, null));
        user.put(KEY_NAME, pref.getString(KEY_NAME, null));
        user.put(KEY_EMAIL, pref.getString(KEY_EMAIL, null));
        user.put(KEY_PASSWORD, pref.getString(KEY_PASSWORD, null));
        return user;}
}
```

Listing 6.2: Extract of Session Manager

6.1.3 Services Implementation

The service package is implemented to provide communication between the application's activities and the web server. As mentioned in the design chapter, JSONParser and ServiceFunctions classes need to be implemented first.

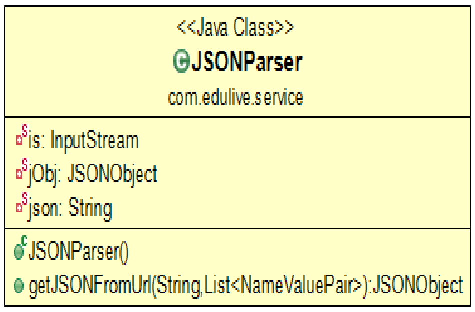


Figure 6.3: JSONParser Class Diagram

Listing 6.3 shows the JSONParser.class code. It contains the getJSONFromUrl (String url, List<NameValuePair> params) method, which accepts as parameters, a URL address type String type and a list of parameters type NameValuePair (HTTP type).

The functionality of this method is to send a request to the given URL address (web server) with the given parameters (TAG, data) and to receive a response from the web server. The final operation is to convert the response to a JSON object [51]. It is decided to convert the response to a JSON object to make it easier for the activities to extract the response data, using the provided JSON methods from the Eclipse IDE.

```
public class JSONParser {
    private static InputStream is = null;
    private static JSONObject jObj = null;
    private static String json = "";
    public JSONObject getJSONFromUrl(String url, List<NameValuePair> params) {
        try {
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new HttpPost(url);
            httpPost.setEntity(new UrlEncodedFormEntity(params));
            HttpResponse httpResponse = httpClient.execute(httpPost);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();
        }
        catch (UnsupportedEncodingException e) { Log.e("JSONParser", "getJSONFromUrl Error:
" + e.getMessage()); }
        catch (ClientProtocolException e) { Log.e("JSONParser", "getJSONFromUrl Error: " +
e.getMessage(); }
        catch (IOException e) { Log.e("JSONParser", "getJSONFromUrl Error: " +
e.getMessage()); }
        try  {
            BufferedReader reader = new BufferedReader(new InputStreamReader(is, "iso-8859-
1"), 8);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "n");
            }
            is.close();
            json = sb.toString(); }
        catch (Exception e) { Log.e("Buffer Error", "Error converting result " +
e.getMessage()); }
        try  {
            jObj = new JSONObject(json);
        }
        catch (JSONException e) { Log.e("JSON Parser", "Error parsing data " +
e.getMessage()); }
      return jObj;  }}
```

Listing 6.3: JSONParser Class

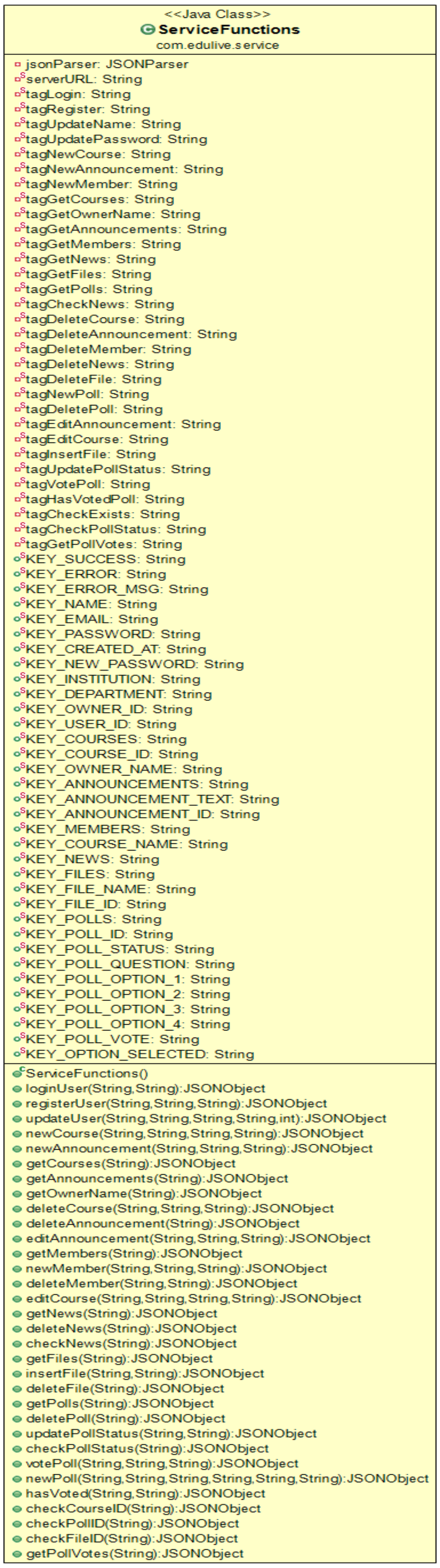

<<Java Class>>
ServiceFunctions
com.edulive.service
jsonParser: JSONParser
serverURL: String
tagLogin: String
tagRegister: String
tagUpdateName: String
tagUpdatePassword: String
tagNewCourse: String
tagNewAnnouncement: String
tagNewMember: String
tagGetCourses: String
tagGetOwnerName: String
tagGetAnnouncements: String
tagGetMembers: String
tagGetNews: String
tagGetFiles: String
tagGetPolls: String
tagCheckNews: String
tagDeleteCourse: String
tagDeleteAnnouncement: String
tagDeleteMember: String
tagDeleteNews: String
tagDeleteFile: String
tagNewPoll: String
tagDeletePoll: String
tagEditAnnouncement: String
tagEditCourse: String
tagInsertFile: String
tagUpdatePollStatus: String
tagVotePoll: String
tagHasVotedPoll: String
tagCheckExists: String
tagCheckPollStatus: String
tagGetPollVotes: String
KEY_SUCCESS: String
KEY_ERROR: String
KEY_ERROR_MSG: String
KEY_NAME: String
KEY_EMAIL: String
KEY_PASSWORD: String
KEY_CREATED_AT: String
KEY_NEW_PASSWORD: String
KEY_INSTITUTION: String
KEY_DEPARTMENT: String
KEY_OWNER_ID: String
KEY_USER_ID: String
KEY_COURSES: String
KEY_COURSE_ID: String
KEY_OWNER_NAME: String
KEY_ANNOUNCEMENTS: String
KEY_ANNOUNCEMENT_TEXT: String
KEY_ANNOUNCEMENT_ID: String
KEY_MEMBERS: String
KEY_COURSE_NAME: String
KEY_NEWS: String
KEY_FILES: String
KEY_FILE_NAME: String
KEY_FILE_ID: String
KEY_POLLS: String
KEY_POLL_ID: String
KEY_POLL_STATUS: String
KEY_POLL_QUESTION: String
KEY_POLL_OPTION_1: String
KEY_POLL_OPTION_2: String
KEY_POLL_OPTION_3: String
KEY_POLL_OPTION_4: String
KEY_POLL_VOTE: String
KEY_OPTION_SELECTED: String
ServiceFunctions()
loginUser(String,String):JSONObject
registerUser(String,String,String):JSONObject
updateUser(String,String,String,String,int):JSONObject
newCourse(String,String,String,String):JSONObject
newAnnouncement(String,String,String):JSONObject
getCourses(String):JSONObject
getAnnouncements(String):JSONObject
getOwnerName(String):JSONObject
deleteCourse(String,String,String):JSONObject
deleteAnnouncement(String):JSONObject
editAnnouncement(String,String,String):JSONObject
getMembers(String):JSONObject
newMember(String,String,String):JSONObject
deleteMember(String,String):JSONObject
editCourse(String,String,String,String):JSONObject
getNews(String):JSONObject
deleteNews(String):JSONObject
checkNews(String):JSONObject
getFiles(String):JSONObject
insertFile(String,String):JSONObject
deleteFile(String):JSONObject
getPolls(String):JSONObject
deletePoll(String):JSONObject
updatePollStatus(String,String):JSONObject
checkPollStatus(String):JSONObject
votePoll(String,String,String):JSONObject
newPoll(String,String,String,String,String,String):JSONObject
hasVoted(String,String):JSONObject
checkCourseID(String):JSONObject
checkPollID(String):JSONObject
checkFileID(String):JSONObject
getPollVotes(String):JSONObject

Figure 6.4: ServiceFunctions Class Diagram

Listing 6.4 shows an extract of ServiceFunctions.class and how it works. As we can see, the class keeps the web server URL address and all keys and the tag that it needs for the method. The constructor creates a JSONParser object which is used in the method. The loginUser() method is used to send an email and a password to the web server and to save the result (whether a user with such details exists or not).

The method accepts parameters type String (email, password) and creates a List<NameValuePair> params in which it creates the parameters that will be sent to the web server (jsonParser.getJSONFromUrl(serverURL, params)). Thus, the first parameter in the list is the (“tag”,”login”), the second parameter is (“email”, email) and the last is (“password”, password). When the jsonParser.getJSONFromUrl(serverURL, params) is executed, these parameters are sent to the web server and a response is saved in the json variable in the loginUser method. The last operation of the method is to return the json response.

ServiceFunctions.class methods are used by the application’s activities to send or download data from/to the web server. The way the web server processes the data will be analysed in Section 6.2.

Listing 6.5 shows an extract of BackgroundService.class and Figure 6.5 the BacgroundService class diagram. BackgroundService.class extends Service.class, which is an application component that can perform long-running operations in the background and does not provide a user interface. It is called as a service (Section 6.1.6) by MainActivity.class to run in the background while the application is running. It is used to check every specific amount of time, if the user has news (has been added to a course). It uses a Handler instance which is associated with a single thread that executes the code which is inside run(). The postDelayed() method is used to set the thread to re-run after the specified amount of time (INTERVAL).

```
public class ServiceFunctions
{
    private JSONParser jsonParser;

    // Service URLS
    private static String serverURL = "http://address/EduLive_API/";

    private static String tagLogin = "login";

    public static String KEY_EMAIL = "email";
    public static String KEY_PASSWORD = "password";


    public ServiceFunctions() { jsonParser = new JSONParser(); }

    public JSONObject loginUser(String email, String password)
    {
        // Building Parameters
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("tag", tagLogin));
        params.add(new BasicNameValuePair(KEY_EMAIL, email));
        params.add(new BasicNameValuePair(KEY_PASSWORD, password));

        // get JSON Object
        JSONObject json = jsonParser.getJSONFromUrl(serverURL, params);

        // return JSON Object
        return json;
    }

}
```

Listing 6.4: Extract of ServiceFunctions

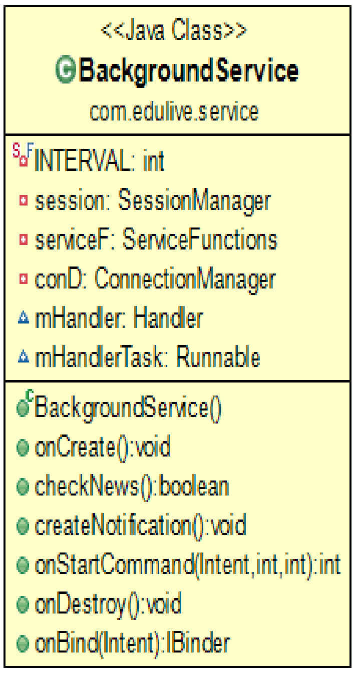
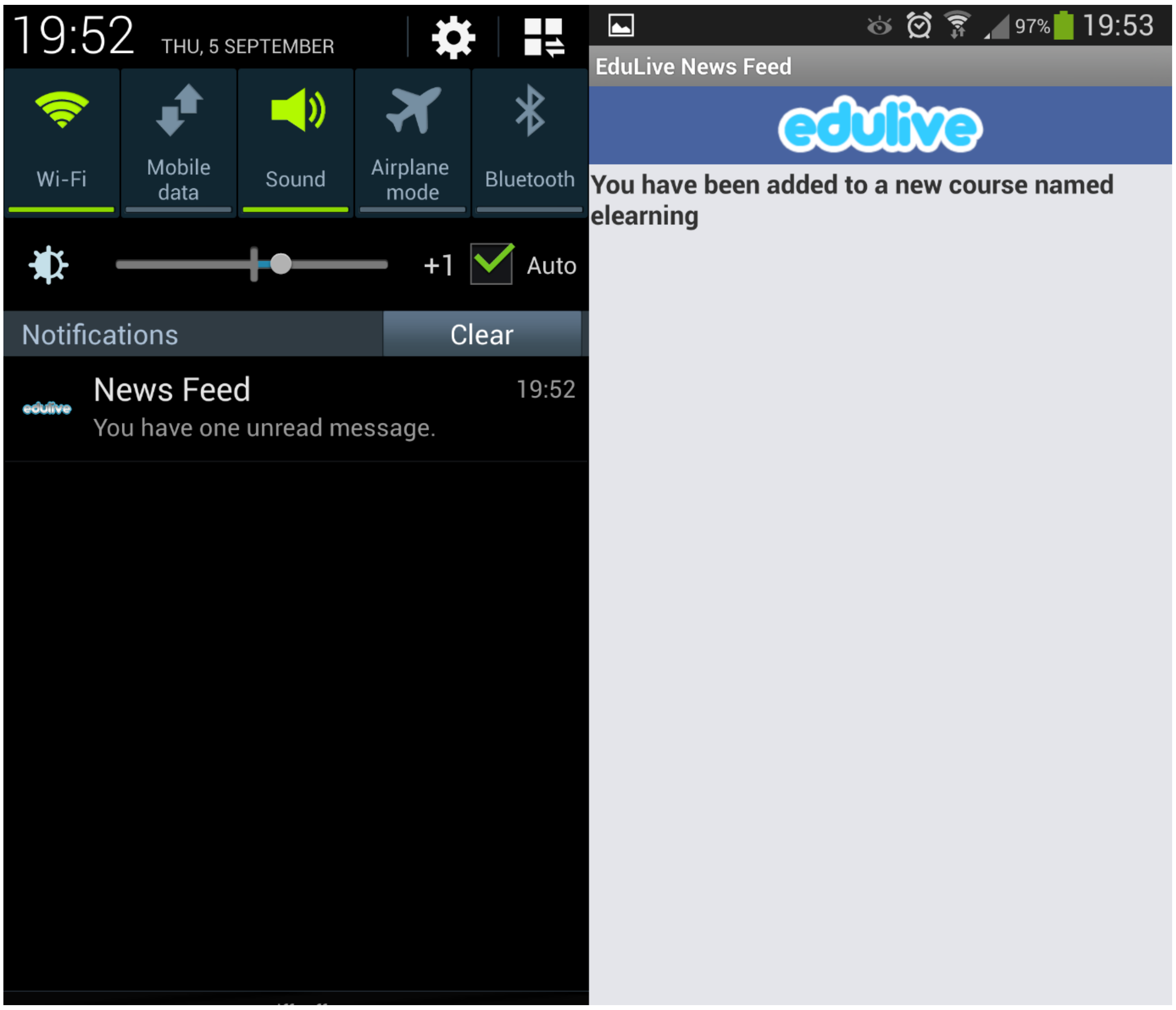


Figure 6.5: BackgroundService Class Diagram

```
Runnable mHandlerTask = new Runnable() {
            @Override
            public void run() {
                boolean hasNews = false;
                try {
                conD = new
ConnectionManager(getApplicationContext());

                if (conD.isNetworkAvailable()) {
                        serviceF = new ServiceFunctions();
                        session = new
SessionManager(getApplicationContext());

                        if (!session.isLoggedIn())
                                stopSelf();

                        hasNews = checkNews();

                        if (hasNews)
                                createNotification();
                }
                }
                catch(Exception e) { stopSelf(); }
                mHandler.postDelayed(this, INTERVAL);
            }
        };
```

Listing 6.5: Extract of BackgroundService

Figure 6.6 shows the notification drawer which is created by the BackgroundService.class to inform the user of a new feed and the NewsActivity screen where the user is redirected after pressing on the notification drawer.



News Feed 19:52

You have one unread message.

EduLive News Feed

You have been added to a new course named elearning

Figure 6.6: News Feed Notification and Screen

6.1.4 GUI Implementation

This section presents the implementation of the user interface design, as it is presented in Section 5.4.

Using Android's XML vocabulary, it is easy to design UI layouts and the screen elements they contain, in the same way you create web pages in HTML. Each layout file must contain one root element (ScrollView, RelativeLayout, LinearLayout). Once the root element is defined, additional layout objects can be added as child elements, to build the whole view which defines the layout. Each layout file must be saved with the .xml extension in the Android project's res/layout/ directory, so it will properly compile.

```
<ScrollView
        android:id="@+id/activity "
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#E4E5EA"
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools" >

       <RelativeLayout
               android:layout_width="match_parent"
               android:layout_height="wrap_content"
               android:background="#E4E5EA"
               tools:context=".Activity" >

         <ImageView
                       android:id="@+id/header"
                       android:layout_width="matchl_parent"
                       android:layout_height="wrap_content"
                       android:background="#4863A0"
                       android:src="@drawable/logo" />

       </RelativeLayout>
</ScrollView>
```

Listing 6.6: Extract of Layout

Listing 6.6 shows an extract of the layout code which is used to design the screens. It declares ScrollView as a root element which provides the scroll feature on the screen. ScrollView contains the background attribute which sets the background of the screen. Inside the root element, a RelativeLayout (or LinearLayout) object, which holds an image object (ImageView) is defined. The image object is the header logo of all screens. The attribute src is the destination of the image file. The "@drawable/logo" means that the image file with the name logo exists in a drawable folder of the project.

In both root and child view object, there are attributes for width and height. They specify the basic height and width of the view. Match_parent is to make the view as big as its parent. Wrap_contents is to make the view enclose its content.

Listing 6.7 shows an extract of the objects that have been used as the child object of the RelativeLayout or LinearLayout in the application's screens. Figure 6.7 shows the result of using the objects of Listing 6.7 in a layout file to create the Poll Creation and the Poll Vote screens. In the Poll Creation screen, the user can fill out the question and answer options and press the submit button to create a new poll. For more options he can press on the Spinner object to change from two to another number. For taking a poll, the user can tick one option from the Poll Vote screen and then press on the submit button.

```
<Button android:id="@+id/button_send"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/button_send"
      android:onClick="sendMessage" />

<EditText
      android:id="@+id/inputSearch"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:hint="@string/search"
      android:layout_below="@id/header"
      android:inputType="textVisiblePassword" />

<ListView
      android:id="@+id/list_view"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:divider="#b5b5b5"
      android:dividerHeight="2dp"/>

<CheckBox
      android:id="@+id/option1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginLeft="50dp"
      android:layout_marginRight="50dp"
      android:layout_marginTop="14dp"
      android:ems="10" />

<TextView
      android:id="@+id/select"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginLeft="50dp"
      android:layout_marginRight="50dp"
      android:layout_marginTop="14dp"
      android:text="Choose an option " />

<Spinner
      android:id="@+id/spinner"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginLeft="50dp"
      android:layout_marginRight="50dp"
      android:layout_marginTop="14dp" />
```

Listing 6.7: Extract of Objects used in Layouts

A description of the defined objects in Listing 6.7 is given below:

Object Name	Description
Button	Is used to create a button that can be pressed, or clicked, by the user to perform an action.
ListView	A view that shows in a vertically scrolling list in all orientations.
CheckBox	An on/off switch which can be toggled by the user.
TextView	Is used to display text to the users and optionally allows them to edit it.
EditText	It is a thin veneer over TextView that configures itself to be editable.
Spinner	It provides a quick way to select one value from a set. Touching the spinner displays a dropdown menu with all other available values, from which the user can select a new one.

Table 6.1: Used Layout Objects

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new "
          android:icon="@drawable/ic "
          android:title="@string/new"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

Listing 6.8: Menu example

In each screen layout we added menus to provide extra functions. Listing 6.8 shows the XML code of a menu with new and help menu items. For each layout, a menu XML file is created in the Android project's res/menu/ directory, so it that it properly compiles. Figure 6.8 shows an example of a menu created in the Course Poll screen which can be used by a user to create a new poll or refresh the polls list or logout from the application.

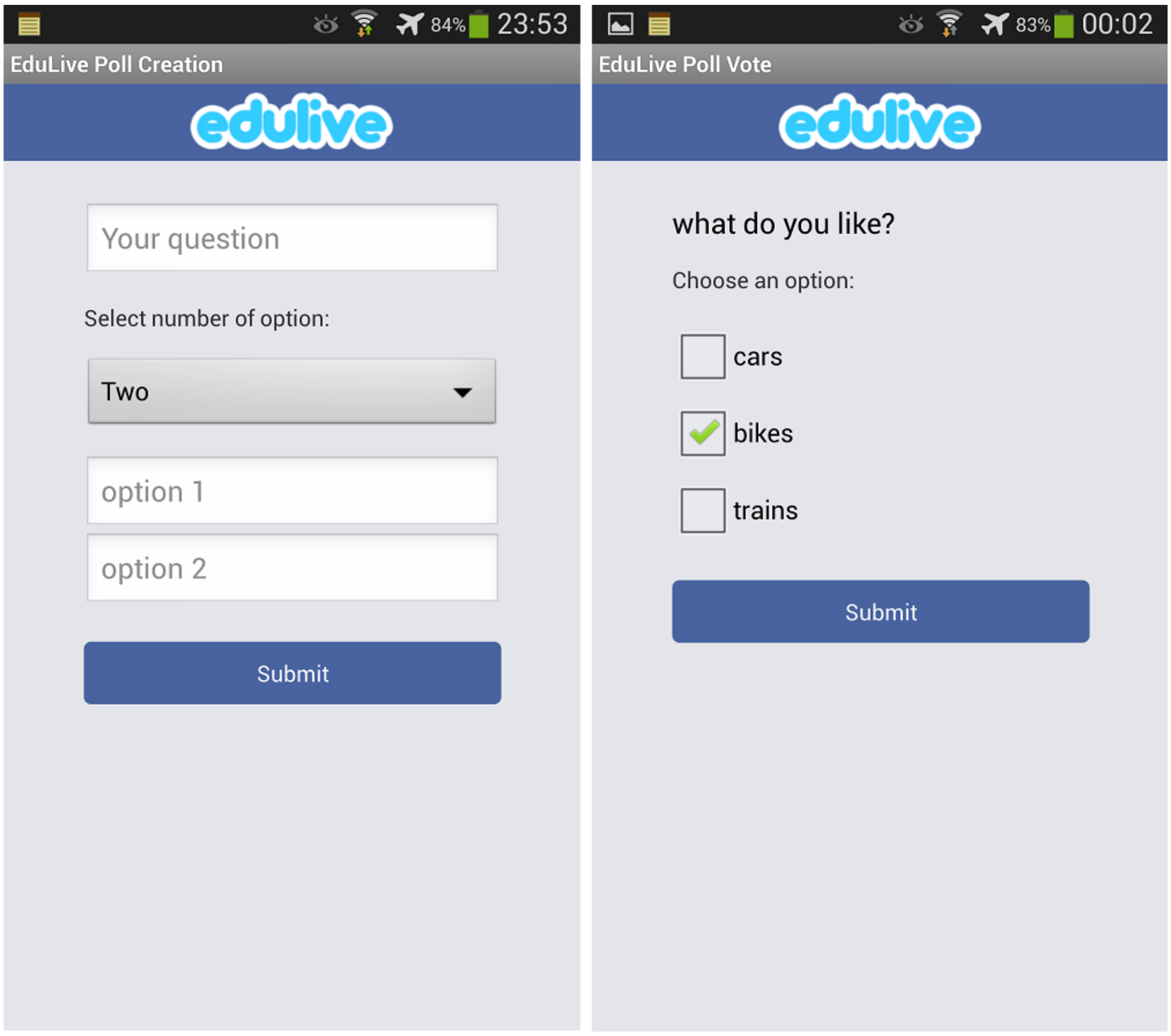


Figure 6.7: Poll Creation and Poll Vote Screen

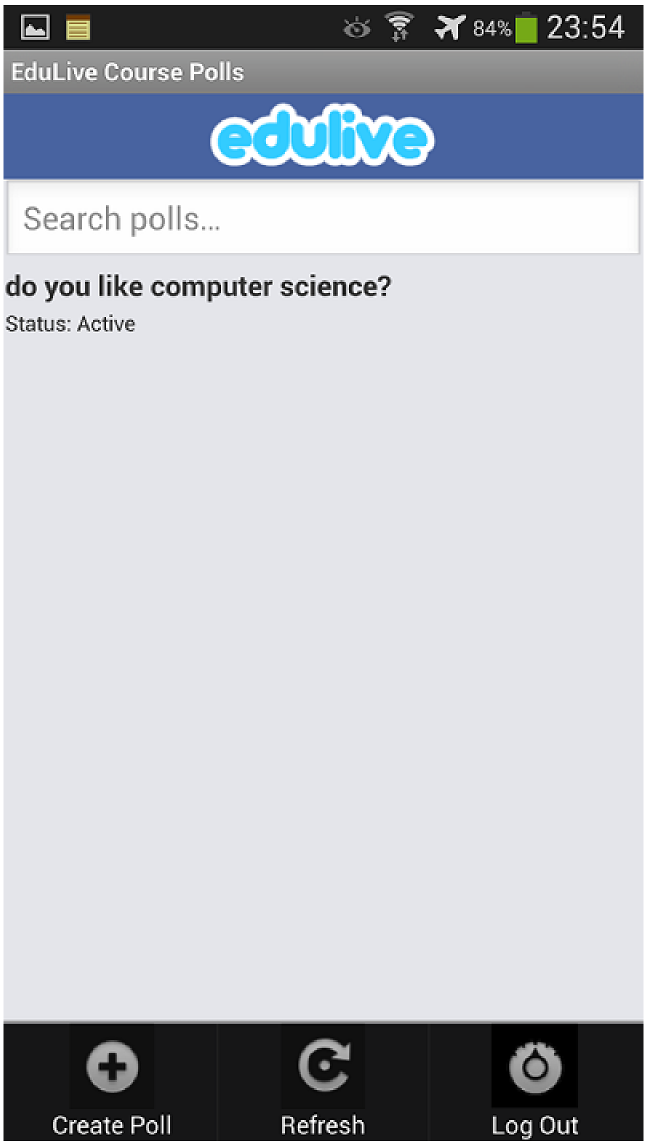


Figure 6.8: Course Polls Screen with Enabled Menu

6.1.5 Activities Main Methods

We start with the implementation of the main methods that have been implemented in all activities. Listing 6.9 and 6.10 show the code of the methods below. A description for these methods is given here:

Methods	Description
onCreate()	It is called when the activity is first created. There all the objects that need to be used should be loaded from the XML file (res/layout/ directory), as it is presented in Section 6.1.4. The setContentView(R.layout.activity_main) method is used to load the layout XML file with name activity_main.
onResume()	It is called just before the activity starts interacting with the user. In the application, it is used only to check if the user is still logged in, otherwise it logs the user out.

onBackPressed()	It is called when the user presses the back key in an activity. The application's activities are terminated when the user presses on the back key, calling the finish() method.
onActivityResult()	It is called when an activity which had launched exits, giving a requestCode. In our application, it is used to pass data from child activity to root activity.
onCreateOptionsMenu()	It is used to load and display the menu from the menu XML file (res/menu/).
onOptionsItemSelected()	It is used to check which Menu item the user pressed on.
onCreateContextMenu()	It is used to create a Context menu, when the activity loads a ListView from an XML file and the user presses on its items.
onContextItemSelected()	It is used to check which Context Menu item the user pressed on.

Table 6.2: Activities Main Methods Description

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
}

@Override
public void onResume()
{
        super.onResume();

        if (!session.isLoggedIn()) { finish();}
}

@Override
public void onBackPressed() { finish(); }

@Override
public boolean onCreateOptionsMenu(Menu menu)
{
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item)
{
        switch (item.getItemId())
        {
                case R.id.logout:
                        return true;
                default:
                        return super.onOptionsItemSelected(item);
        }
}
```

Listing 6.9: Activities' Main Methods

Figure 6.9 shows an example of the Context Menu and its options in the Course Polls Screen, while a user is trying to access a poll. Pressing on Open will redirect the user to the Poll Vote screen and pressing on View Results will redirect the user to the Poll Results screen. If the user is the owner of the course, he will be able to see the Lock and Remove Context Menu item. The Lock option can lock the poll, disabling anyone from voting. The Remove option removes the poll from the list of all the course members.

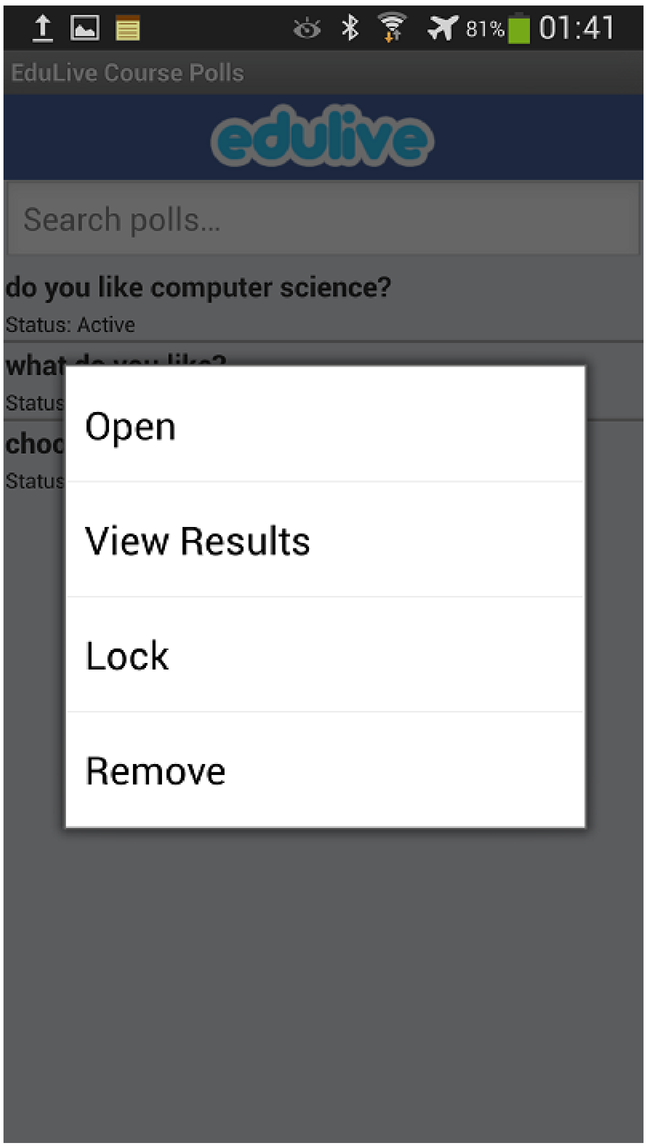


Figure 6.9: Context Menu in Course Polls Screen

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
                if (requestCode == 1) //after the return of course activity
                {
                        if (resultCode == 4)
                                finish();
                }
}


@Override
public boolean onContextItemSelected(MenuItem item)
{
        AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
        final int position = (int) info.id;

        switch(item.getItemId())
        {
                case 1:
                return true;
                case 2:
                return true;
                default:
                return super.onContextItemSelected(item);
              }
        }
 }


@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo)
{
        super.onCreateContextMenu(menu, v, menuInfo);
        AdapterContextMenuInfo info = (AdapterContextMenuInfo) menuInfo;
        int position = info.position;
        menu.add(Menu.NONE, 1, Menu.NONE, "Open");
        menu.add(Menu.NONE, 2, Menu.NONE, "Info");

}
```

Listing 6.10: Activities' Main Methods

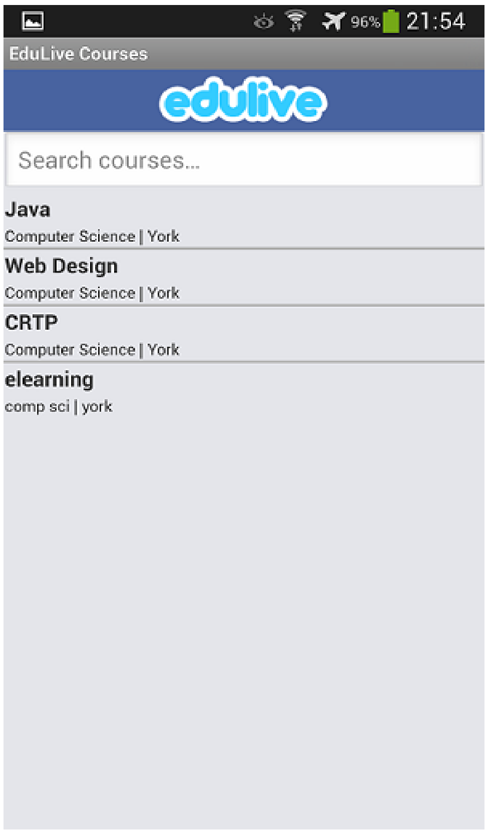
6.1.6 Using Layout Objects in Activities

This section presents how the activities are using the layout objects.

The first three lines of Listing 6.11 show how the layout objects are loaded to variables. In order to add a click listener to a button, a setOnClickListener must be used. Thus, when a user presess on the button, the code inside the brackets of the onClick() method will be executed.

```
TextView question = (TextView) findViewById (R.id.question);
ListView list = (ListView) findViewById(R.id.list_view);
Button profileButton = (Button) findViewById(R.id.btnProfile);


profileButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {}
});
```

Listing 6.11: Loading Layout Objects and Set onClickListener()



Java
Computer Science | York
Web Design
Computer Science | York
CRTP
Computer Science | York
elearning
comp sci | york

Figure 6.10: Courses List

Listing 6.12 shows how data from an external source is used to create a ListView that represents each data entry. Thus, in our application, the data from the web server is retrieved in a data variable (type List<Map<String, String>>) and then using a SimpleAdapter, they are displayed (list.setAdapter(adapter)) in the application's screen as a list.

Figure 6.10 shows the Courses Screen, which uses a ListView and a SimpleAdapter to display the courses (item of the list) and the courses' institution and department (sub item of the list). Pressing on a course will redirect the user to the Courses Screen which, contains the course's Map (Polls, Announcements, Files, and Members) (Figure 5.10).

```
SimpleAdapter sa = new SimpleAdapter(this, data, R.layout.list_item,
                       new String[] {"item", "subitem"}, new int[] {R.id.item, R.id.subitem});
list.setAdapter(sa);
```

Listing 6.12: Filling a ListView with data using Simple Adapter

6.1.7 Activities Using other Activities and Libraries

So far, it has been mentioned what implementations have been made on the GUI, service, manager, and activities side. Below, it is presented how all the above implementations can be combined and run together as one application.

Firstly, we need to define how an activity can start other activities or services (Listing 6.13) for a user to view other screens layouts. The listing below shows how an activity starts CourseActivity and starts BackgroundService as a service. For example, if a user is in the main screen and wants to view the course screen by pressing a button, the code below should be added in an action listener (Listing 6.11).

```
startActivity(new Intent(getApplicationContext(), CoursesActivity.class));
startService(new Intent(this, BackgroundService.class));
```

Listing 6.13: Starting an Activity and a Service

Secondly, we need to define how the activities can have access to the web server. Listing 6.14 shows how an activity tries to access the web server and receive the courses data. A ConnectionManager, ServiceFunctions and the ProgressDialog object is created.

The ProgressDialog object is used to show a loading dialog to inform the user that the activity is processing data. A Thread is created to run in parallel and execute the code for retrieving the web server's data. ProgressDialog is cancelled when the Thread finishes its processing.

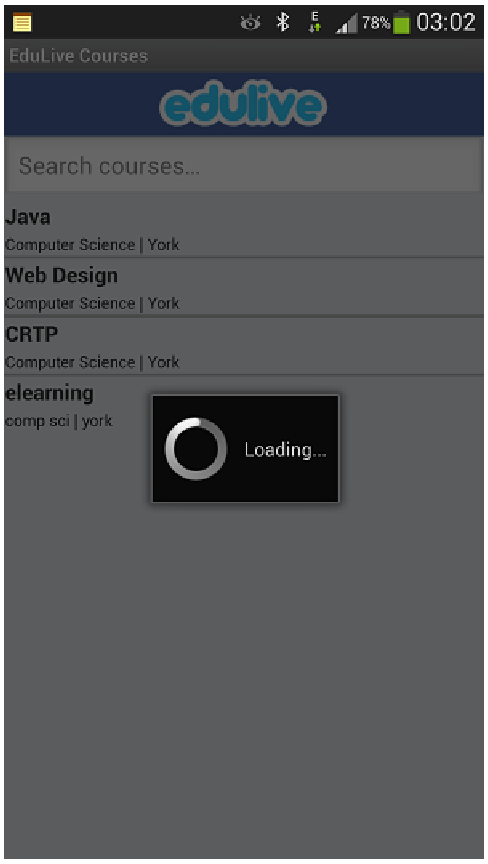


Figure 6.11 ProgressDialog while retrieving Courses

The ConnectionManager object is used to check if there is access to the internet and, if this is true, the ServiceFunctions object calls the getCourses() method to retrieve the data in a JSON format. The extractData() method is used in the listing to show that the system uses the JSON object to extract the data from it.

Figure 6.11 shows ProgressDialog (Loading message) when the activity tries to retrieve the courses.

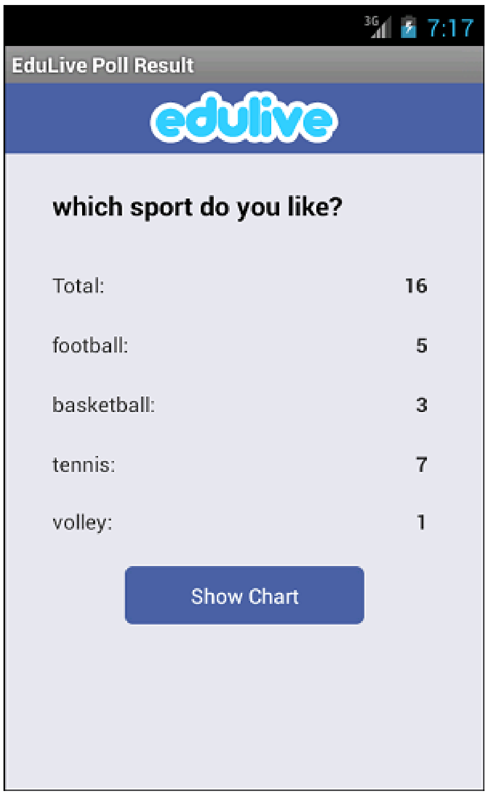
Thirdly, we need to define how an activity can use SessionManager and CourseSessionManager to save or load data (Listing 6.15). Listing 6.15 shows the creation of a SessionManager and a CourseSessionManager object and the use of their methods createCourseSession() and getUserDetails(). Using the createCourseSession() method the course details are stored in CourseSessionManager. Using getUserDetails(), we load the data of the user who is logged in, in a HashMap.

```
ConnectionManager conD = new ConnectionManager(getApplicationContext());
ServiceFunctions serviceF = new ServiceFunctions();
ProgressDialog pDialog;

pDialog = ProgressDialog.show(.this, "", "Loading...", true);
new Thread (new Runnable() {
        public void run() {
                runOnUiThread(new Runnable() {
                public void run() {
                      if (conD.isNetworkAvailable()) {
                              JSONObject json = serviceF.getCourses(uID);
                              extractData(json);
                              pDialog.cancel();
                        }
                }
           });
        }}).start();
```

Listing 6.14: Activity Retrieves Web Server's Data

```
SessionManager session = new SessionManager(getApplicationContext());
CourseSessionManager sessionCourse = new CourseSessionManager(getApplicationContext());
sessionCourse.createCourseSession("name", "owner", "id");
HashMap<String, String> details = session.getUserDetails();
```

Listing 6.15: Use of SessionManager and CourseSessionManager



EduLive Poll Result

which sport do you like?

Total: 16

football: 5

basketball: 3

tennis: 7

volley: 1

Figure 6.12: Poll Result Screen

Finally, we need to define how an activity can use an external library. The Android SDK makes it easy to add other libraries (jars), which can then be used from the activities by just importing their file path (org.achartengine.*).

In our project, we are using AChartEngine [54] to display the poll's results in a chart. Figure 6.12 shows the Poll Result Screen. Pressing on the Show Chart Button, we are redirected to the Poll Result Chart Screen (Figure 6.13), which shows the same results in a bar chart.

Listing 6.16 shows the implementation of the Poll Result Chart screen using the AChartEngine library classes. In order to create the screen we need to draw the columns, creating XYSeriesRenderer objects and then to add them to a XYMultipleSeriesDataset object which is the background area of the chart. Then, we use ChartFactory.getBarChartIntent() which is a utility method for creating chart intent to start it as an activity. String votes_X is the number of votes for each option_X in the poll. (X = 1,..,4)

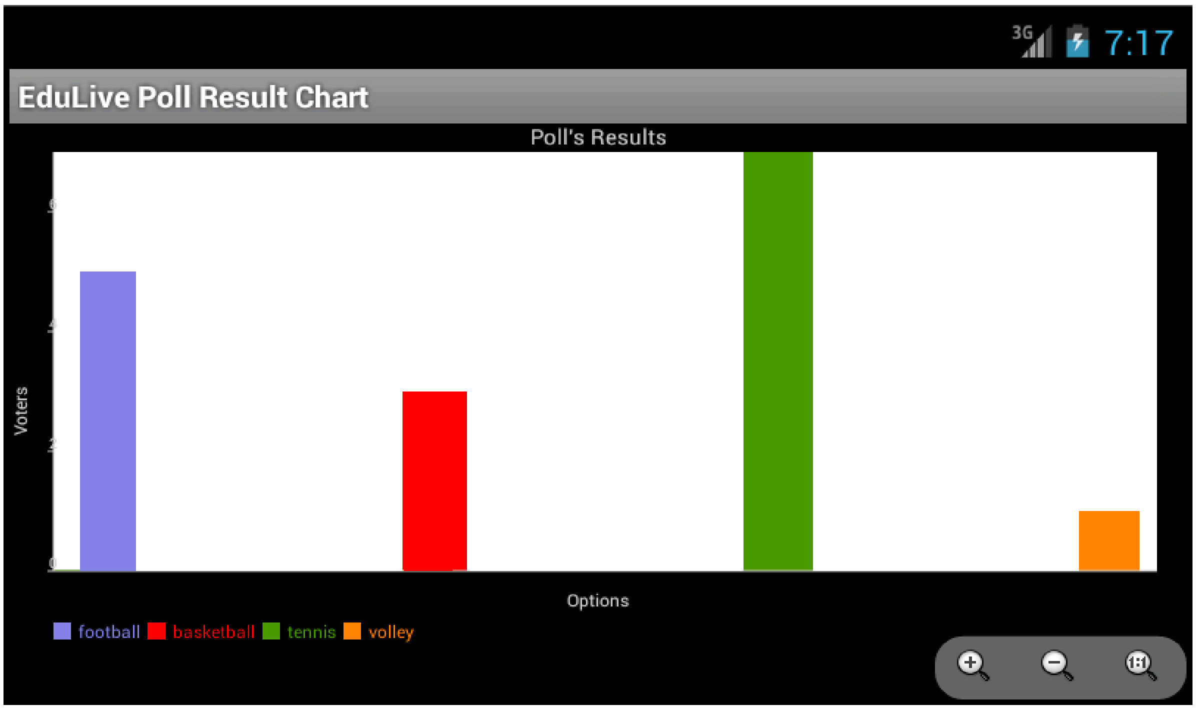


Figure 6.13: Poll Result Chart Screen

```
        String votes_1, votes_2, votes_3, votes_4,option_1, option_2, option_3, option_4;
        // Creating an  XYSeries for options
        XYSeries option1Series = new XYSeries(option_1);
        XYSeries option2Series = new XYSeries(option_2);
        XYSeries option3Series = new XYSeries(option_3);
        XYSeries option4Series = new XYSeries(option_4);
        // Adding data to option Series
        option1Series.add(0,0);
        option2Series.add(0,0);
        option3Series.add(0,0);
        option4Series.add(0,0);
        option1Series.add(3,Integer.parseInt(votes_1));
        option2Series.add(1,0);
        option3Series.add(2,0);
        option4Series.add(4,0);
        option1Series.add(5,0);
        option2Series.add(6,Integer.parseInt(votes_2));
        option3Series.add(7,0);
        option4Series.add(8,0);
        option3Series.add(9,Integer.parseInt(votes_3));
        option1Series.add(10,0);
        option2Series.add(11,0);
        option4Series.add(12,Integer.parseInt(votes_4));
        option1Series.add(14,0);
        option2Series.add(15,0);
        option3Series.add(16,0);
        // Creating a dataset to hold each series and adding to them the series
        XYMultipleSeriesDataset dataset = new XYMultipleSeriesDataset();
        dataset.addSeries(option1Series);
        dataset.addSeries(option2Series);
        dataset.addSeries(option3Series);
        dataset.addSeries(option4Series);
        // Creating XYSeriesRenderer to customize option1 Series
        XYSeriesRenderer option1Renderer = new XYSeriesRenderer();
        option1Renderer.setColor(Color.rgb(130, 130, 230));
        option1Renderer.setFillPoints(true);
        option1Renderer.setLineWidth(60);
        option1Renderer.setDisplayChartValues(false);
        // Creating XYSeriesRenderer to customize option2 Series
        XYSeriesRenderer option2Renderer = new XYSeriesRenderer();
        option2Renderer.setColor(Color.rgb(255, 0, 0));
        option2Renderer.setFillPoints(true);
        option2Renderer.setLineWidth(60);
        option2Renderer.setDisplayChartValues(false);
        // Creating XYSeriesRenderer to customize option3 Series
        XYSeriesRenderer option3Renderer = new XYSeriesRenderer();
        option3Renderer.setColor(Color.rgb(76, 153, 0));
        option3Renderer.setFillPoints(true);
        option3Renderer.setLineWidth(60);
        option3Renderer.setDisplayChartValues(false);
        // Creating XYSeriesRenderer to customize option4 Series
        XYSeriesRenderer option4Renderer = new XYSeriesRenderer();
        option4Renderer.setColor(Color.rgb(255, 128, 0));
        option4Renderer.setFillPoints(true);
        option4Renderer.setLineWidth(60);
        option4Renderer.setDisplayChartValues(false);
        // Creating a XYMultipleSeriesRenderer to customize the whole chart
        XYMultipleSeriesRenderer multiRenderer = new XYMultipleSeriesRenderer();
        multiRenderer.setXLabels(0);
        multiRenderer.setChartTitle("Poll's Results");
        multiRenderer.setXTitle("Options");
        multiRenderer.setYTitle("Voters");
        multiRenderer.setZoomButtonsVisible(true);
        // Adding options Renderer to multipleRenderer
        multiRenderer.addSeriesRenderer(option1Renderer);
        multiRenderer.addSeriesRenderer(option2Renderer);
        multiRenderer.addSeriesRenderer(option3Renderer);
        multiRenderer.addSeriesRenderer(option4Renderer);
        Intent chart = ChartFactory.getBarChartIntent(getBaseContext(), dataset, multiRenderer,
Type.DEFAULT);
            startActivity(chart);
```

Listing 6.16: Poll's Results Chart Screen Implementation

6.2 Web Server Implementation

This section presents the implementation of the web server. Based on the design described in Chapter 5, the code written is presented.

In Section 5.3.1, it is mentioned that the web server consists of four PHP files. In this section the implementation of these files is presented.

Listing 6.17 shows the extract of the index PHP file. The HTTP POST request is received by the index file, which checks if the request contains a tag. If a tag doesn't exist or if it cannot be recognized, the index file sends the message "Access Denied" or "Invalid Request" respectively as a response. If a tag exists, it saves its content to a variable $tag, which is checked in a list of if-else statements. The if-else statements check what the tag contains. Also, the index file creates a DB_Functions object which will be used for the database's processing inside the if-else statements' brackets.

Listing 6.17 presents the check for the "login" tag. If the request contains the tag login it means that the mobile application is trying to login using the user's input data. Then, the email and password from the request are extracted to $email and $password variables. These variables are used in the $db->getUserByEmailAndPassword method which returns an array of data. The array of data is added to the $response array with the success tag set to 1 and then it is returned (response) as a JSON object. If the return of the DBFunction method is false (empty data) a JSON response is returned with the error tag set to 1 and with the error message in an error_msg key.

Listing 6.18 shows the DB_Connect and DB_Functions extract and the uploadFile PHP files respectively. DB_Connect uses the PHP MySQL functions to connect to the database and returns the database handler. DB_Functions uses DB_Connect to execute the MySQL statements.

In the extract of DB_Functions, the implementation of getUserByEmailAndPassword is presented to complement the index implementation extract. The method executes a MySQL query which selects from the users table the user with the given name and email ($email, $password). If the result table has one row, the method returns it, otherwise it returns false.

That way, the index file which called this method returns the return array as a JSON result or an error message if the getUserByEmailAndPassword returned false.

Lastly, the uploadFile implementation is presented, which stores the temporary uploaded file to a new location ($filepath). The temporary file disappears when the script ends.

```php
<?php
if (isset($_POST['tag']) && $_POST['tag'] != '')
{
    $tag = $_POST['tag'];
    require_once 'include/DB_Functions.php';
    $db = new DB_Functions();

    $response = array("tag" => $tag, "success" => 0, "error" => 0);

    if ($tag == 'login')
    {
        $email = $_POST['email'];
        $password = $_POST['password'];
        $user = $db->getUserByEmailAndPassword($email, $password);

        if ($user != false)
        {
            $response["success"] = 1;
            $response["user"]["uid"] = $user["uid"];
            $response["user"]["name"] = $user["name"];
            $response["user"]["email"] = $user["email"];
            echo json_encode($response);
        }
        else
        {
            $response["error"] = 1;
            $response["error_msg"] = "Incorrect email or password!";
            echo json_encode($response);
        }
    }
    else
    {
        echo "Invalid Request";
    }
}
else
{
    echo "Access Denied";
}
?>
```

Listing 6.17: Extract of index PHP file

```
<?php
class DB_Connect
{
    public function connect()
    {
        $con = mysql_connect(“localhost”, “user”, “password”);
        mysql_select_db(“database”);
        return $con;
    }
}?>
<?php
class DB_Functions
{
   $private $db;
    function __construct()
    {
        require_once 'DB_Connect.php';
        $this->db = new DB_Connect();
        $this->db->connect();
     }
public function getUserByEmailAndPassword($email, $password) {
        $result = mysql_query("SELECT * FROM users WHERE email = '$email'") or
die(mysql_error());
        $no_of_rows = mysql_num_rows($result);
        if ($no_of_rows > 0)  {
            $result = mysql_fetch_array($result);
            if ($result['password'] == base64_encode(pack('H*', sha1($password)))) { return
$result; }
        }
        else {  return false; }
    } ?>
<?php
    $file_path = "uploads/";
    $file_path = $file_path . basename( $_FILES['uploaded_file']['name']);
    if(move_uploaded_file($_FILES['uploaded_file']['tmp_name'], $file_path)) {
        echo "success"; } else { echo “fail” } ?>
```

Listing 6.18 Extract of DB_Connect and DB_Functions

Chapter 7: Evaluation & Testing

This chapter presents the evaluation and testing of the project. The first two sections present the testing of the functional requirements using the Android framework and the non-functional requirements evaluation respectively. The third section presents the application's UI testing using the Android tools, followed by the application's evaluation. The last section is a summary of all the results of the testing that has taken place.

7.1 Functional Requirements Testing

In order to test if the application meets the defined functional requirements in Section 4.2.1, we used the integrated testing framework of the Android framework and the SDK tools to set up and run the tests [55]. Thus, we tested all aspects of the application during and after the implementation.

The following tables present the test cases written to evaluate the correct functionality of the application. Each test case has a unique identifier, followed by the functional requirement it tests, the test input, and the pass criteria.

Test Reference	TS-01
Tested Requirement	FR01
Test Content	Checks that the application allows the user to sign up.
Input	User fills out the sign up form, using his name, email and password.
Pass Criteria	Application displays a progress dialog and then redirects the user to the main screen.

Test Reference	TS-02
Tested Requirement	FR02
Test Content	Checks whether the system allows user to log in.
Input	Users fills out the login form using his email and password.
Pass Criteria	Application displays a progress dialog and then redirects the user to the main screen.

Test Reference	TS-03
Tested Requirement	FR05
Test Content	Checks that the application allows user to change his email and password after he is logged in.
Input	Users fills out the new email and password dialog adding the old, new email and password.
Pass Criteria	Application shows a progress dialog and redirects user to the profile screen.

Test Reference	TS-04
Tested Requirement	FR08
Test Content	Checks that the application displays the user's courses in a list when user is navigating in courses screen.

Input	The user presses on Courses button.
Pass Criteria	Application displays a progress dialog and then displays the user's courses in a list.

Test Reference	TS-05
Tested Requirement	FR20
Test Content	Checks that the application allows user to view the poll and then allows him to vote in it.
Input	User presses on a poll, opens it, chooses an option and presses on the submit button.
Pass Criteria	Application displays the context menu to the user with the Open option, then redirects user to the poll's screen displaying the question and the available options. When user presses on the submit button, systems show a message "Thank you for your vote"

Test Reference	TS-06
Tested Requirement	FR15
Test Content	Checks that the application allows user to view the course's files and then to download them.
Input	User presses on the Files button and then presses on a file from the Files list.
Pass Criteria	Application displays the course's files in a list and then displays a progress dialog showing the progress of the download in a percentage bar. When the download is finished, the file is located in the SDcard path.

The other test cases are available in Appendix C.

Table 7.1 presents the test results of the test cases. The error recognition messages during the login, sign up and change user's details (name, password) processes have been checked in one test case (TS-08). All the functional requirements were met during the implementation and thus all test cases passed.

Test Reference	**Test Result**	**Test Reference**	**Test Result**
TS-01	**PASS**	TS-14	**PASS**
TS-02	**PASS**	TS-15	**PASS**
TS-03	**PASS**	TS-16	**PASS**
TS-04	**PASS**	TS-17	**PASS**
TS-05	**PASS**	TS-18	**PASS**
TS-06	**PASS**	TS-19	**PASS**
TS-07	**PASS**	TS-20	**PASS**
TS-08	**PASS**	TS-21	**PASS**
TS-09	**PASS**	TS-22	**PASS**
TS-10	**PASS**	TS-23	**PASS**
TS-11	**PASS**	TS-24	**PASS**
TS-12	**PASS**	TS-25	**PASS**
TS-13	**PASS**	TS-26	**PASS**

Table 7.1: Functional Requirements Test Cases Result

7.2 Non-Functional Requirements Evaluation

In this section, it is checked whether the non-functional requirements of the application which were defined in Section 4.2.2 were successfully fulfilled.

- **NFR01:** The application offers the same user interface in all screens using the same style for menus, buttons and layouts as they are presented in the design and implementation chapters.

- **NFR02:** The application should show clear and detailed notification messages to the user. This was tested by navigating through all the application's screens and using all the functions which have been implemented. For every action a user performs, the system shows a detailed Toast message (explained in Section 5.4.4).

- **NFR03:** The application must have a lack of bugs and must inform the user of any wrong operation. This was tested by the produced test cases in the previous section.

- **NFR04:** The application should be able to run on all Android devices. This is fulfilled by the implementation of the application using the Android SDK in Eclipse IDE. Every Android project which is extracted to an .apk file can be run on all Android devices (Section 2.4.2).

- **NFR05:** The application should request a password for each user account. Nobody can have access to the main screen of the application without passing through the sign in operation. If the user adds blank data to the password field, the system will display an error recognition message, as it was tested in the test cases in the previous section. The same goes for the sign up process which displays the same messages when the user leaves the password fields empty.

- **NFR06:** The application should have a fast response time. This was tested by using all the available features of the application. The threads that were used in the implementation for retrieving the web server's data minimised the application's response time.

- **NFR07:** The application should be designed with the ability to accept new operations and features. The design of the application helps accept new operations and features by adding new activities or modifying the existing ones. A new operation can be added by creating new methods in the current activities or adding new activities. New activities need only use the service and manager package for accessing the web server.

7.3 Testing the Application's UI

In addition to unit testing the individual components that make up the application (such as activities, services, and content providers), it is also important to test the behaviour of the application's user interface (UI) when it is running on a device. UI testing ensures that the application returns the correct UI output in response to a sequence of user actions on a device. [56]

We followed two approaches for testing the UI during the implementation phase and after. The first approach was to run tests manually on an Android device to verify that the application is behaving as expected. The second approach was to automate the UI testing with a software testing framework. An automated test involves creating programs to perform testing tasks to cover specific usage scenarios (test cases), and then using the testing framework to run the test cases automatically and in a repeatable manner [56].

To run the automated test cases we used the provided Android SDK tools: [56]

- uiautomatorviewer: a GUI tool to scan and analyse the UI components of an Android application.
- uiautomator: a Java library containing APIs to create customized functional UI tests, and an execution engine to automate and run the tests.

The following table presents UI test cases that evaluate the correct UI output of the application. Each test case has been tested following both a manual and an automated approach.

Test Reference	Test Content	Test Result
TSUI-01	Checks that the application UI shows the keyboard input in all EditText objects (such as name, password, title, text, email forms).	**PASS**
TSUI-02	Checks that the application UI shows the ListView objects in all screens with ListView objects (Courses, Files, Members, and Announcements).	**PASS**
TSUI-03	Checks that the application UI shows the Menu in all screens.	**PASS**
TSUI-04	Check that the application UI shows all buttons correctly in all screens.	**PASS**
TSUI-05	Checks that the application shows the Context Menus while pressing the ListView items in every screen with a ListView object.	**PASS**
TSUI-06	Checks that the application shows the Progress Dialogs in every screen which has access to the web server.	**PASS**
TSUI-07	Checks that all screens have lack of crashing UI errors.	**PASS**
TSUI-08	Checks that the application UI shows the header image correctly in all screens.	**PASS**
TSUI-09	Checks that the application UI shows the Main Screen's image buttons.	**PASS**
TSUI-10	Checks that the application UI shows the error recognition messages in the colour red in the EditText objects.	**FAIL**
TSUI-11	Checks that the application UI shows the Toast messages correctly in all screens.	**PASS**
TSUI-12	Checks that the application UI shows all objects in all orientations.	**PASS**

Table 7.2: UI Test Cases

Almost all UI test cases passed. The test case (TSUI-10) failed the manual testing on the testing device, unlike the automated test. The error recognition message could be displayed, but not in the colour red which is automatically provided by Android, when an EditText is set to be displayed with an error. This error was fixed by adding colour to the error messages using SpannableStringBuilder which is provided by Android SDK.

7.4 Application's Evaluation

In the previous sections we discussed how we tested and evaluated the application from the viewpoint of the designer/developer. In this section, we present the application's evaluation from the viewpoint of the user.

There are two expert-based evaluation techniques that can confirm whether the application meets its requirements:

- **Heuristic evaluation** – is a method for quick, cheap, and easy evaluation of the user interface design. The goal of the heuristic evaluation is to find the usability problems in the design so that they can be attended to as part of an interactive design process. The heuristic evaluation involves having a small set of evaluators examine the interface and judge its compliance with recognized usability principles (the "heuristics"). [57]

- **Cognitive walkthrough evaluation –** is a usability inspection method used to identify usability issues in interactive systems. It is task-specific, whereas the heuristic evaluation takes a holistic view in order to spot problems not spotted by this and other usability inspection methods. The method is rooted in the notion that users typically prefer to learn a system by using it to accomplish tasks, rather than, for example, studying a manual. The method is prized for its ability to generate results quickly with low cost, especially when compared to usability testing, as well as for the ability to apply the method early in the design phases, before coding has even begun. [58][59]

The application's development was completed within the vacation period, so it was impossible to run it in a class with a teacher and students as evaluators. Thus, the cognitive walkthrough evaluation was chosen to be done by an expert evaluator, because the heuristic evaluation requires three to five evaluators and cannot replace an actual user evaluation [60].

7.4.1 Expert Evaluation

An E-Learning Adviser from the Academic support office of the University of York was chosen and requested to perform a cognitive walkthrough evaluation.

A description of the application was given to the evaluator along with the tasks (use cases Figure 4.1) in Table 7.3, on an Android mobile device and tablet.

Task	Steps Followed
Sign up	**1.** Press on the register link of the starting screen **2.** Fill out the form **3.** Press on the Register button
Sign in	**1.** Press on the register link of the starting screen **2.** Fill out the form **3.** Press on the Register button
Create a course	**1.** Press on courses button of the main screen **2.** Press on the menu button **3.** Press on the new button **4.** Fill out the form and press confirm

Invite a member	**1.** Select a course from the course list **2.** Press on the members button **3.** Press on the menu button **4.** Press on the add member button **5.** Fill out the form and press add
Create an announcement	**1.** Select a course from the courses list **2.** Press on the announcements button **3.** Press on the menu button **4.** Press on create announcement **5.** Fill out the form and press add
Upload a file	**1.** Select a course from the courses list **2.** Press on the files button **3.** Press on the menu button **4.** Press on upload **5.** Find the file from the file manager and press on it
Download a file	**1.** Select a course from the courses list **2.** Press on the files button **3.** Press on file
Create a poll	**1.** Select a course from the course list **2.** Press on the polls button **3.** Press on the menu button **4.** Press on create **5.** Fill out the form **6.** Press on the submit button
Vote in a poll	**1.** Select a course from the course list **2.** Press on the polls button **3.** Press on a poll **4.** Press on open **5.** Choose an option **6.** Press on the submit button
View the poll's results	**1.** Select a course from the course list **2.** Press on the polls button **3.** Press on a poll **4.** Press on result **5.** Press on the chart button to view the results in a chart
Delete a course	**1.** Long press on a course from the course list **2.** Press on delete **3.** Press on the confirm button
Delete a member	**1.** Long press on a member from the members list **2.** Press on delete **3.** Press on the confirm button
Delete a file	**1.** Long press on a file from the files list **2.** Press on delete **3.** Press on the confirm button
Delete a poll	**1.** Long press on a poll from the polls list **2.** Press on delete **3.** Press on the confirm button
Delete an announcement	**1.** Long press on an announcement from the announcements list

	2. Press on delete **3.** Press on the confirm button

Table 7.3: Steps Followed to do the Tasks

The expert evaluator didn't find any bug or error in the application. This was expected, as the functional requirements and UI testing were done before the evaluation.

Evaluator's notes on application's strengths:

- Simple, easy to use GUI
- Students can create and administer courses
- Flexible enrolment
- Polls easy to create, respond to and view the results

Evaluator's general notes

The evaluator believes that the application has the potential to support both face to face teaching and independent study away from contact time. The ability to create "courses" that can be used to share files and to create announcements that are restricted to a predefined group of students who need to be enrolled to be able to access, mirrors the simple functionality of many standard virtual learning environments (VLEs) [61] or learning management systems (LMSs) [62].

He mentions that a number of mobile apps and services already exist for major VLEs that come with the significant advantage that they utilise existing online course delivery, removing the need for manual course creation and enrolment as well as avoiding duplicating content delivery and providing students with multiple places to check for course content. They are also very feature-rich, offering academics significantly more pedagogic options than the project's application, such as online quizzing, discussion boards, video delivery etc.

Many of the functions could also potentially be delivered through Google apps and groups.

The application's course features might be most suitable for situations or training providers without access to an LMS platform, or those wishing to create simple ad hoc mobile groups, where references to other sources of online support is not required.

The application's poll feature is potentially more applicable in a face to face classroom setting, acting as an audience response system (ARS). There are a number of these type of services currently on the market, where the emphasis lies on simplicity of use for instructors and students. One big issue with all applications currently on the market is the disassociation between live classroom interaction and other online provision (e.g. student performance in classroom interactions is not recorded with other student grades held in the VLE).

Evaluator's suggestions for areas development

The evaluator believes that the application is well realised, given the development resources available. He also mentions that there is a danger of it trying to replicate existing functionality that is better supported or has closer integration with other widely used systems, such as institutional VLEs and student record systems.

Potential areas for development could include:

- Simplifying course creation and management workflow.
- Simplifying polling participation workflow, so it can be delivered without need for account creation and courses.
- Integration between application and currently used VLEs / LMSs – be aware of danger of replicating existing mobile application functionality.

7.5 Summary

The goal of this Chapter was to present all the tests and evaluations that took place during the implementation phase and afterwards to evaluate the correct functionality of the application.

All test cases for the functional requirements were passed which means that the application has no functional error. The non-functional requirements were met too. In order to test the UI we followed two approaches (manual testing in a device and automated testing using Android SDK tools). One test failed in the manual approach, but it was a minor problem caused by the specific device and it was fixed in the implementation phase. After the functional and UI testing, the application was tested by an expert (E-adviser) who did a cognitive walkthrough evaluation. During the walkthrough evaluation, all use cases were tested. In the end of the evaluation, the evaluator mentioned the strengths of the application and wrote a summary about similar applications and technologies. Last but not least, suggestions for further areas for development were given.

Chapter 8: Conclusions

This chapter summarizes the work done in this project, followed by the project's contribution and future work.

8.1 Work Summary

The goals of the project were to study the benefits of m-learning, to provide an analysis of principles and patterns of mobile interface design, and tactics that solve common mobile development problems. Lastly, its goal was to develop a mobile application for the Android platform that would provide functions which support distance learning and offer direct communication between students and their teachers through the internet. The goals were accomplished, since a study was made on m-learning (Chapter 2), and mobile interface design principles, patterns and development tactics (Chapter 3). The ultimate goal was accomplished as well, as all the functional and non-functional requirements were met (Chapter 7).

At the end of the work summary section, we want to refer to the agile development model which was used in the Android mobile application development. In Section 5.2.1, it is mentioned that Android software development doesn't follow a structural design or pattern, so it is difficult for developers to work on an Android application. The agile model allows us to easily change the requirements when needed, and the frequent small incremental releases of the software help us quickly improve the application's design, add more features and fix the bugs. We believe that the agile model is the best development model for developing an Android application, but it needs to be stressed that it requires experience in developing Android applications.

8.2 Project Contribution

The contribution to the education and mobile developers' community is described in this section. The contribution is addressed below:

- A study and analysis of m-learning and its theory was made:
    - Key features were analysed.
    - The factors that create learning incentives in m-learning were analysed.
    - Current perspectives and valuable elements were analysed.
    - The benefits and the arguments for/against the m-learning were defined.

  Also, the study of m-learning in this report can positively alter the perceptions of parents of a mobile generation consisting of students of all ages. It can also highlight the need to engage students in mobile learning during and beyond the school days and pave the way for schools to welcome and not prohibit mobile technologies and social networking.

- Design principles and patterns of mobile interface design were analysed. Guidelines and instructions were given, which can be followed by new mobile developers for designing and developing good applications. This analysis may also help developers understand and consider a few important things when they want to design a new mobile application or transfer a full-sized computer application to a mobile environment.

- Study of the Android platform was made, and presented and its architecture was analysed.

- Investigation of tools that would be used to design and develop a mobile application in the Android operating system was made. The design and development of an Android application requires the Eclipse IDE and its ADT plugin to be able to use the Android SDK.

- Tactics for solving common problems in Android application development to help new developers were provided.

- A fully functional Android application was created, which can support distance learning. The main advantages of creating this application are:
    - It is designed to make students more active and motivated in lectures.
    - Currently, there are not many applications that provide distance learning or that can be used during lectures.
    - As it is an Android application, it can be installed for free in all Android devices (mobile phones and tablets), by being shared in the Google Store.
    - The source can be used by other developers to extend the current functions or add new ones that will enhance the education lectures.

- Evaluation of the application was made. The evaluation was based on unit testing for the functional requirements and UI. An expert evaluation took place to evaluate the usefulness of the application and suggest new areas for development.

8.3 Future work

This section discusses the possible improvements that can be made in the future to improve the application and add new functions.

Improvements in the navigation

The GUI of the application offers a minimal and easy navigation through the application's screens. As a result, the user needs to go to the main screen to navigate from the main to other screens (courses and news feed) and from the course map to other screens (polls, announcements, members and files). To avoid this, future developers can implement a navigation drawer. The navigation drawer is a panel that transitions in from the left edge of the screen and displays the application's main navigation options. It has the ability to access any top level content from anywhere in the application, no matter how deep the level a user is in.

Improvements in the file manager

The file manager which was implemented provides browsing only by the screen. For example, if the user wants to go to a previous folder, he must press on the "../" symbol. Pressing the device's back button closes the file manager. It is trivial for future developers to enable the back functionality while keeping the file manager on and direct the user to the previous folder.

Replace the login and remove registration process

The registration and login process is time-consuming for a user. As a result, it is very trivial for future developers to use the device's primary email address for the login purpose and remove the registration process.

Simplify the course creation and management workflow

As mentioned by the evaluator in the previous chapter, the course creation and management workflow could be simplified. A new approach for this is, instead of inviting users to a course, the owner of a course to share a course identification number, which is generated during the course creation. Future developers have to customize the web server and ServiceFunctions.class to provide and accept a unique identification number respectively, and the courses' screen to provide a field for joining a course using an id.

Simplify the polling participation workflow

As mentioned by the evaluator in the previous chapter, the polling participation workflow could be simplified. Future developers can change the polls screen from a sub screen of a course to a main screen similar to the courses screen and follow the approach which is defined in the previous paragraph. Thus, users can access a poll using an identification number, which has been shared by the owner of the poll. Developers also need to modify the web server response to provide an identification number and ServiceFunctions.class.

Add privileges option

The current implementation allows only one owner per course. Future developers could add an option for adding owner privileges to course participants. Developers have to add this function to the web server files, ServiceFunctions.class and modify the courses' owner_id field in the database to accept an array of ids.

Add more chart types

The current chart implementation provides only a column chart. Future developers could add more options, such as bar charts, line charts, pie charts, area charts, bubble charts and doughnut charts. Developers have to modify the showChart() method in PollResultActivity.class.

8.4 Concluding Remarks

Having mentioned both the theoretical and the practical aspects of mobile application development, as well as its primary uses and benefits concerning education, and more specifically, m-learning, we can provide an overall account of our study. To begin with, in our project we analysed the software development process followed for Android applications, as well as the design principles, patterns and tactics which are useful for mobile application development. As we note, also important for the successful creation and maximum utilization of our application is the theory of m-learning, as the understanding of its principles and elements is important for developing a beneficial education tool.

Having taken all these factors into consideration, we put them into practice and created an m-learning Android application, suitable for use both in and outside the classroom, as it provides the ability for students and teachers to communicate and exchange educational material through the use of the application from their mobile devices. The application thus provides significant benefits to the education field, as it enables constant mobile learning, which enhances the educational experience, making it ubiquitous. To conclude, we believe that, with further improvement from future developers, the already useful application we created could become fully operational within the education system, as it has much to offer by greatly adding to and evolving the educational process.

References

[1] D. Randy Garrison, *E-Learning In the 21st Century: A Framework for Research and Practice*, 2nd ed. New York: Routledge, 2011.

[2] T. Georgiev, E. Georgieva, and A. Smrikarov, "M-Learning – a new Stage of E-Learning," in International Conference on Computer Systems and Technologies, 2004.

[3] Software development process. (2013). [Online]. Available: http://www.computerhope.com/jargon/s/softdeve.htm

[4] Ian Sommerville, *Software Engineering*, 8th ed. Essex, England: Pearson Education Limited, 2007.

[5] What is Waterfall model – advantages, disadvantages and when to use it?. (2013). [Online]. Available: http://istqbexamcertification.com/what-is-waterfall-model/

[6] What is Incremental model – advantages, disadvantages and when to use it?. (2013). [Online]. Available: http://istqbexamcertification.com/what-is-incremental-model/

[7] What is Spiral model – advantages, disadvantages and when to use it?. (2013). [Online]. Available: http://istqbexamcertification.com/what-is-spiral-model/

[8] What is Agile model – advantages, disadvantages and when to use it?. (2013). [Online]. Available: http://istqbexamcertification.com/what-is-agile-model/

[9] M. Sharples, J. Taylor, and G.Vavoula, "Towards a Theory of Mobile Learning," in proceedings of mLearning Conference, Cape Town, 2005.

[10] G. N. Vavoula, "A Study of Mobile Learning Practices," Internal report of MOBIlearn project, 2005.

[11] B. Banks. (2004). *E-Portfolios: Their Uses and Benefits*. [Online]. Available: http://archive.excellencegateway.org.uk/media/fd%20learning/e-Portfoliopaper.pdf

[12] G. Vavoula, "KLeOS: A Knowledge and Learning Organisation System in Support of Lifelong Learning," Unpublished Ph.D dissertation, University of Birmingham, UK, 2004.

[13] M. Sharples, "Disruptive devices: mobile technology for conversational learning," *International Journal of Continuing Engineering Education and Life Long Learning*, 12(5/6), 2002, pp. 504-520.

[14] M. Sharples, "Learning As Conversation: Transforming Education in the Mobile Age," in Proceedings of Conference of Seeing, Understanding, Learning in the Mobile Age, Budapest, Hungary, 2005.

[15] L.S. Vygotsky, *Mind in society: the development of higher psychological*

Processes, Cambridge: Harvard University Press, 1978.

[16] L. Hickman, *John Dewey's Pragmatic Technology*, Bloomington: Indiana University Press, 2003.

[17] H. Kynäslahti, "In Search of Elements of Mobility in the Context of Education," in *Mobile Learning*, Helsinki, 2003, pp. 41-48.

[18] Marcus Boyes. (2011, December 17). *24 Benefits of mobile learning* [Online]. Available: http://insights.elearningnetwork.org/?p=507

[19] D. Belshaw (2010). *Outline and Scope: What is 'mobile learning'* [Online]. Available: http://mobilereview.jiscpress.org/2010/11/2-ii-what-is-mobile-learning/

[20] D. Singelee, B. Preneel, "The Wireless Application Protocol (WAP)*," COSIC Internal Report, 2013.

[21] Otto Peters, *Learning and Teaching in Distance Education: Analyses and Interpretations from an International Perspective*, London, Kogan Page, 2001.

[22] M. Sharples, "The design of personal mobile technologies for lifelong learning", in *Journal of Computers & Education*, University of Birmingham, Oxford UK, 2000, pp. 177-193.

[23] J. Waycott, "The appropriation of PDAs as learning and workplace tools: an activity Theory perspective," Ph.D dissertation, The Open University, Milton Keynes, UK.

[24] L. F. Motiwalla, "Mobile learning: A framework and evaluation", in *Journal Computers & Education*, 2007, vol. 49, pp. 581-596.

[25] M. Sharples, "Big Issues in Mobile Learning", Report of a workshop by the Kaleidoscope Network of Excellence Mobile Learning Initiative, 2006.

[26] J. Wishart, A. McFarlane, and A. Ramsden, "Using personal digital assistants (PDAs) with internet access to support initial teacher training in the UK", University of Bristol, UK, 2005.

[27] M. Sharples, "How can we address the conflicts between personal informal education and traditional classroom education". In: Sharples, M. (Ed.). Big Issues in Mobile Learning, Nottingham, UK, 2006.

[28] M. Milrad, "How should learning activities using mobile technologies be designed to support innovative educational practices?," Nottingham, UK, 2006.

[29] M. Sharples, "The design of personal mobile technologies for lifelong learning," in *Journal of Computers and Education*, 2000, vol. 34, pp. 177–193.

[30] J. Stark. (2012, March 15). *Principles of Mobile Interface Design* [Online]. Available: http://www.slideshare.net/jonathanstark/principles-m-interface-design

[31] B. Ballard, *Designing the Mobile User Experience*, Wiley, 2007.

[32] Benjamin Speckmann, "The Android mobile platform," *A Review Paper submitted to the Eastern Michigan University*, 2008.

[33] Android. (2013). [Online]. Available: http://www.android.com/

[34] N. Smyth, "An Overview of the Kindle Fire Android Architecture," *Kindle Fire App Development Essentials*, 2013, ch. 7, pp. 49- 53.

[35] Exploring the SDK. (2013). [Online]. Available: http://developer.android.com/sdk/exploring.html

[36] Android SDK. (2013). [Online]. Available: http://developer.android.com/sdk/index.html

[37] Android SDK Manager. (2013). [Online]. Available: http://developer.android.com/tools/help/sdk-manager.html

[38] Eclipse Foundation. (2013). [Online]. Available: http://www.eclipse.org/downloads/

[39] Eclipse Public License – v.1.0. [Online]. Available: http://www.eclipse.org/legal/epl-v10.html

[40] Eclipse Documentation. (2013). [Online]. Available: http://www.eclipse.org/documentation/

[41] ADT Plugin. (2013). [Online]. Available: http://developer.android.com/tools/sdk/eclipse-adt.html

[42] Android DDMS. (2013). [Online]. Available: http://developer.android.com/tools/debugging/ddms.html

[43] Reza B'Far, *Mobile Computing Principles: Designing and Developing Applications with UML and XML*, Press Syndicate of the University of Cambridge, Cambridge, United Kingdom, 2005.

[44] T. Neil, *Mobile Design Pattern Gallery: UI Patterns for Mobile Applications*, O'Reilly Media, 2012.

[45] Mary Weilage. (2011, August 10). *10 things I hate about developing for Android* [Online]. Available: http://www.techrepublic.com/blog/10-things/

[46] A. B. Cremers, Sascha Alda, *Non-functional Requirements* [Online]. Available: http://www.iai.uni-bonn.de/slides/10_Interactive%20Systems.pdf

[47] Ian F. Alexander, Neil Maiden, *Scenarios, Stories, Use Cases: Through the Systems Development Life Cycle*, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, England, 2004.

[48] Hypertext Transfer Protocol, RFC 2616. [Online]. Available:

http://tools.ietf.org/html/rfc2616/

[49] PHP. (2013, August 16). [Online]. Available: http://www.php.net/

[50] MySQL. (2013). [Online]. Available: http://www.mysql.com/

[51] JSON (JavaScript Object Notation). (2013). [Online]. Available: http://www.json.com/

[52] James Deruvo. (2011, April 11). *Android is a developers mess according to a survey* [Online]. Available: http://androidcommunity.com/android-is-a-developers-mess/

[53] UML 2 Sequence Diagrams. [Online]. Available: http://www.agilemodeling.com/artifacts/sequenceDiagram.htm

[54] AChartEngine. (2013). [Online]. Available: http://www.achartengine.org/

[55] Android WorkFlow Testing. (2013). [Online]. Available: http://developer.android.com/tools/testing/index.html

[56] Android UI Testing. (2013). [Online]. Available: http://developer.android.com/tools/testing/testing_ui.html

[57] Jakob Neilsen. (1995, January 1). *Ten Usability Heuristics* [Online]. Available: http://www.nngroup.com/articles/ten-usability-heuristics/

[58] Clayton Lewis and John Rieman. (1994). *Evaluating the Design Without Users: Cognitive Walkthroughs*. [Online]. Available: http://hcibib.org/tcuid/chap-4.html

[59] C. Wharton et al. “The cognitive walkthrough method: a practitioner's guide,” in *J. Nielsen & R. Mack “Usability Inspection Methods”*, 1994, pp. 105-140.

[60] Jakob Neilsen. (1995, January 1). *How to Conduct a Heuristic Evaluation* [Online]. Available: http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

[61] Jennifer Styron. (2008, July 21). *Virtual learning environments* [Online]. Available: http://www.slideshare.net/tigerjenn11/virtual-learning-environments

[62] Elis, Ryann K. (2009). *Field Guide to Learning Management System* [Online]. Available: http://www.astd.org/~/media/Files/Publications/LMS_fieldguide_20091

Appendix A

Use Case Scenarios

In this chapter the use case scenarios for the rest of the functional requirements are given.

Use Case Name	Edit profile
Context	Edit profile info (password, name)
Primary Actor	Course Participant
Precondition	Application is running, user is logged on
Success Post Condition	Profile was updated correctly
Trigger	User wants to update password or name
Main Success Scenario	**1.** User press on profile icon button to be redirected in profile window **2.** User press on edit icon next to name or password **a.** If name edit icon is pressed user adds the new name in pop up window and press confirm **b.** If password edit icon is pressed user adds the old password, new password and retypes the new password in pop up window and then press confirm **3.** System closes the pop up window if the update of the profile info is correctly **a.** If there is a problem with the internet connection an error message is displayed to the user **Exception** **b.** System redirects user to profile window

Use Case Name	Create course
Context	Create a new course
Primary Actor	Course Participant
Precondition	Application is running, user is logged on
Success Post Condition	New course was added correctly
Trigger	User wants to create a new course
Main Success Scenario	**1.** User press on courses icon button to be redirected to courses window **2.** User press on new icon button **3.** System displays a dialog form window **4.** User adds a name, institution, department in dialog window and then press confirms **5.** System closes dialog if the course has been added and shows message about the action **a.** If course exists a message is displayed to user **Exception**

	b. If there is a problem with the internet connection an error message is displayed to the user **<u>Exception</u>**

Use Case Name	Add users
Context	Add user(s) to course
Primary Actor	Course Owner
Precondition	Application is running, user is logged on, user selected a course
Success Post Condition	Users added to the course
Trigger	Invite user(s) to a course
Main Success Scenario	**1.** User press on add button **2.** System displays a dialog form **3.** Users add email(s) to add other user(s) **4.** System closes the dialog if the users are added on the course **a.** If the email(s) do not exist a window with the wrong emails is displayed **<u>Exception</u>** **b.** If something goes wrong with the service a message is displayed **<u>Exception</u>** **c.** If there is a problem with the internet connection or with the filled out operation an error message is displayed to the user **<u>Exception</u>**

Use Case Name	Share a file
Context	Share a file with course's members
Primary Actor	Course Owner
Precondition	Application is running, user is logged on, user selected a course and user is owner of this course
Success Post Condition	A file has been uploaded correctly on the webserver
Trigger	Share a file with course's members
Main Success Scenario	**1.** User press on share file button **2.** Use chooses a file from the device **3.** System shows a progress message **a.** If the file has been uploaded correctly a dialog is displayed **b.** If there is a problem with the internet connection or with the upload operation an error message is displayed to the user **<u>Exception</u>**

Use Case Name	View poll results
Context	View poll results
Primary Actor	Course Participant
Precondition	Application is running, user is logged on, user selected a course, user pressed on polls
Success Post Condition	User views the poll results correctly
Trigger	View poll results

Main Success Scenario	**1.** User choose to view results from a poll **2.** Systems show a dialog with the poll's results **a.** If there is a problem with the internet connection or with the viewing operation an error message is displayed to the user **Exception**

Use Case Name	Add announcement
Context	Add an announcement in the course
Primary Actor	Course Owner
Precondition	Application is running, user is logged on and user selected a course and user is owner of this course
Success Post Condition	New announcement was added correctly
Trigger	User wants to add a new announcement
Main Success Scenario	**1.** User press on announcements icon button to be redirected to announcements window **2.** User press on new icon button **3.** System displays a dialog form window **4.** User adds a text and then press confirms **5.** System closes dialog if the announcement has been added and shows message about the action **a.** If announcement with this name exists, a message is displayed to user **Exception** **b.** If there is a problem with the internet connection or with the adding operation an error message is displayed to the user **Exception**

Use Case Name	Leave course
Context	Leave from a course
Primary Actor	Course Participant
Precondition	Application is running, user is logged on, user selected the course
Success Post Condition	Course was removed from the course lists
Trigger	User wants to leave a course
Main Success Scenario	**1.** User press on courses icon button to be redirected to courses window **2.** User selects the course and press on leave course **3.** System displays a dialog window with cancel and confirm buttons **4.** User press on confirm **5.** System closes dialog and deletes the course from the user's courses list If there is a problem with the internet connection or with the adding operation an error message is displayed to the user **Exception**

Use Case Name	Download a file
Context	Download a file from the course
Primary Actor	Course Participant
Precondition	Application is running, user is logged on, user selected the course and then the file
Success Post Condition	File was downloaded
Trigger	User wants to download the file
Main Success Scenario	**1.** User press on courses icon button to be redirected to courses window **2.** User selects the course and press on files icon **3.** User selects the file and press download **4.** File is downloaded and saved in the mobile device If there is a problem with the internet connection or with the downloading operation an error message is displayed to the user **Exception**

Use Case Name	Edit/Delete course
Context	Edit/Delete course
Primary Actor	Course Owner
Precondition	Application is running, user is logged on, user selected the course
Success Post Condition	Course Edited/deleted from all participants and owner
Trigger	User wants to edit/delete one of his courses
Main Success Scenario	**1.** User press on courses icon button to be redirected to courses window **2.** User presses on edit/delete **a.** If edit, system shows the form dialog to edit the details of the course. If pressed confirm the course details is updated in all participants. **b.** If delete, systems show the deleted dialog. If user pressed confirm the course is deleted from user and all participants. If there is a problem with the internet connection or with the editing/deleting operation an error message is displayed to the user **Exception**

Use Case Name	Remove participant
Context	Remove participant from the course
Primary Actor	Course Owner
Precondition	Application is running, user is logged on, user selected the course
Success Post Condition	User removes a participant from the course
Trigger	Users want to remove a participant from the course
Main Success Scenario	**1.** User press on courses icon button to be redirected to courses window **2.** User selects the course and press on participants **3.** User chooses the participant and press remove

	4. System shows the confirmation dialog. **a.** If user pressed confirm participant is removed from the course and cannot access it. If there is a problem with the internet connection or with the removing participant operation an error message is displayed to the user **<u>Exception</u>**

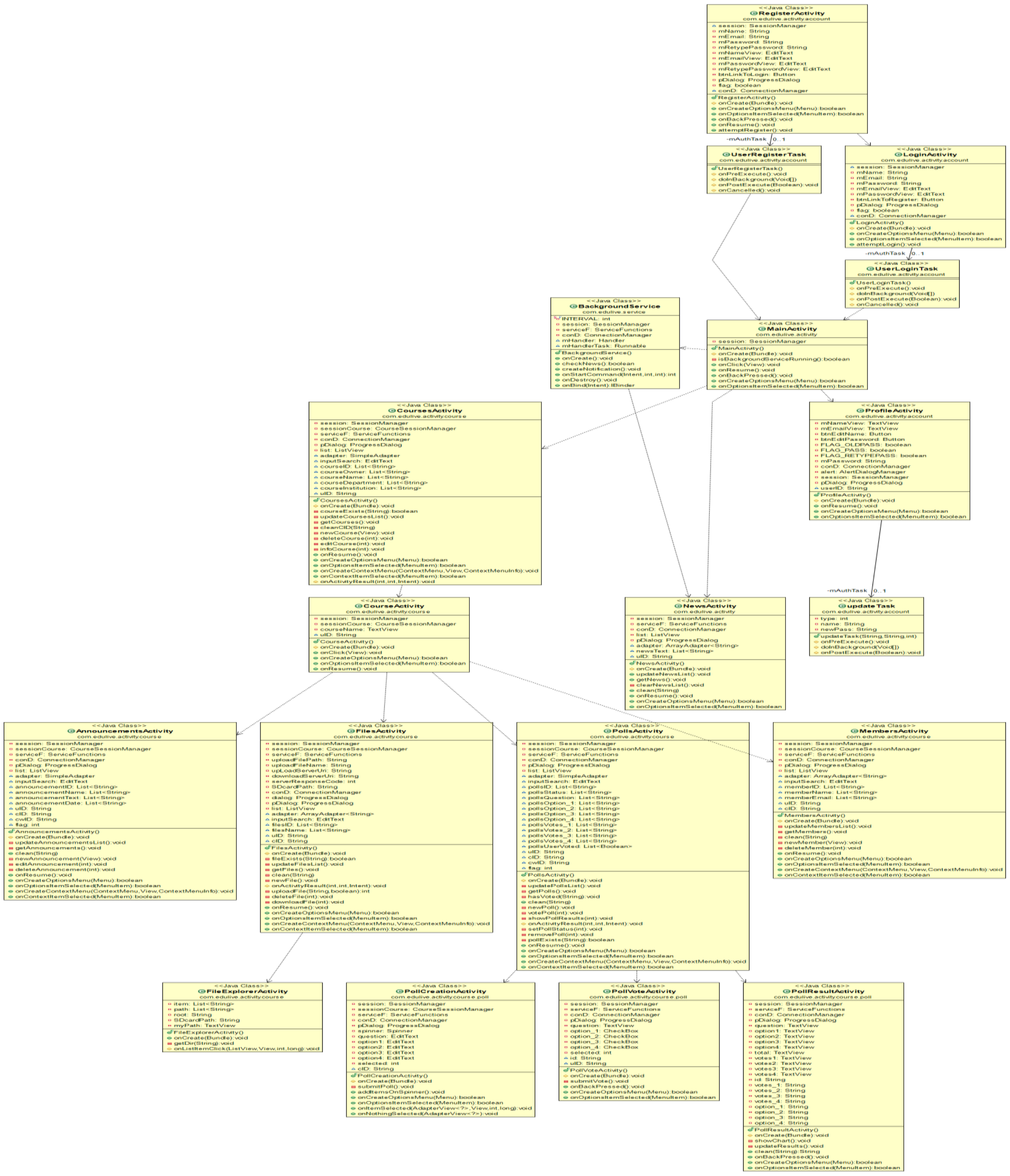
Use Case Name	Edit/Delete announcement
Context	Edit/Delete announcement
Primary Actor	Course Owner
Precondition	Application is running, user is logged on, user selected the course's announcement
Success Post Condition	Announcement is edited/deleted
Trigger	User wants to edit/delete announcement
Main Success Scenario	**3.** User press on courses icon button to be redirected to courses window **4.** User selects the course and press on announcements **5.** User presses on edit/delete **c.** If edit, system shows the form dialog to edit the context of the announcement. If pressed confirm the course context is updated in all participants and owner **d.** If delete, systems show the remove dialog. If user pressed confirm the announcement is removed from user and all participants. If there is a problem with the internet connection or with the editing/removing announcement operation an error message is displayed to the user **<u>Exception</u>**

Use Case Name	Delete a file
Context	Delete a file from the course
Primary Actor	Course Owner
Precondition	Application is running, user is logged on, user selected the course and then the file
Success Post Condition	File was deleted
Trigger	User wants to delete the file
Main Success Scenario	**1.** User press on courses icon button to be redirected to courses window **2.** User selects the course and press on files icon **3.** User selects the file and press delete **4.** File is being deleted If there is a problem with the internet connection or with the deleting operation an error message is displayed to the user **<u>Exception</u>**

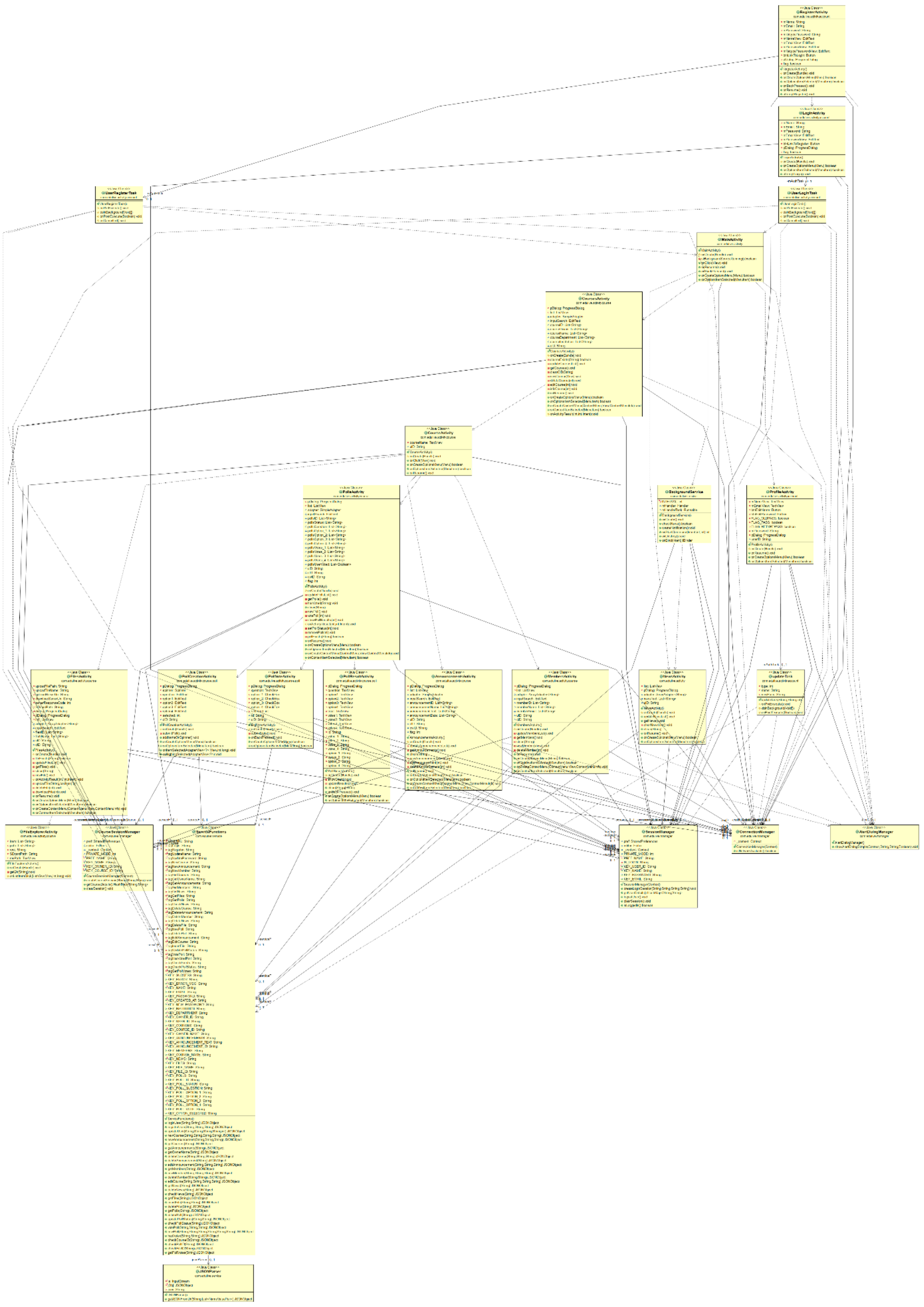
Use Case Name	Delete poll
Context	Delete poll
Primary Actor	Course Owner
Precondition	Application is running, user is logged on, user selected the course and then the poll
Success Post Condition	Poll is deleted
Trigger	User wants to delete poll
Main Success Scenario	**1.** User press on courses icon button to be redirected to courses window **2.** User selects the course and then selects the poll **3.** User presses on delete **a.** Systems show the remove dialog. If user pressed confirm the poll is removed from user and all participants. If there is a problem with the internet connection or with the removing poll operation an error message is displayed to the user **Exception**

Appendix B

Extended Class Diagrams

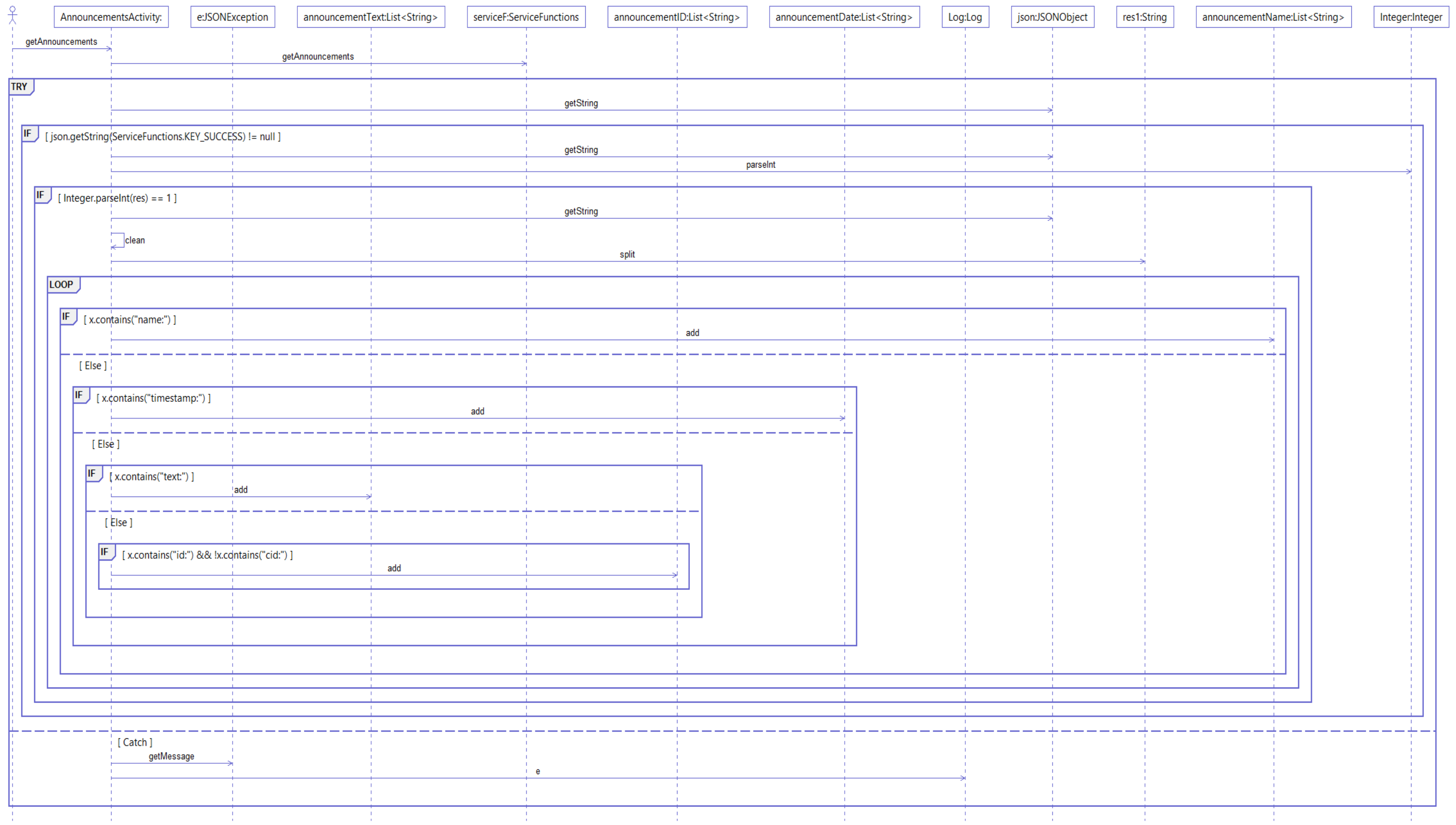


Appendix B.1: Extended Class Diagram



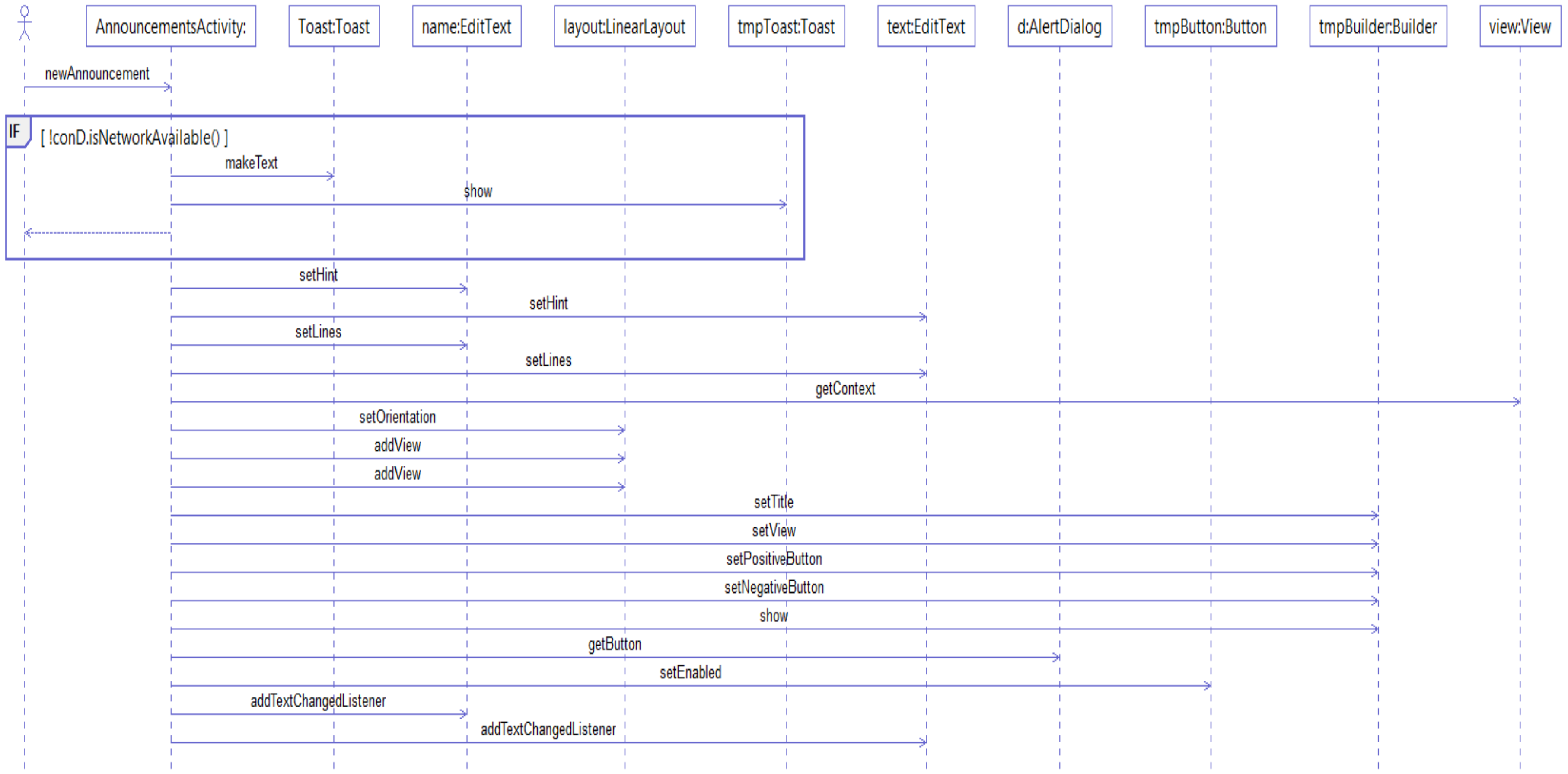
Appendix B.2: ***Extended Class Diagram with manager package and JSONParser, ServiceFunctions classes***

Sequence Diagrams

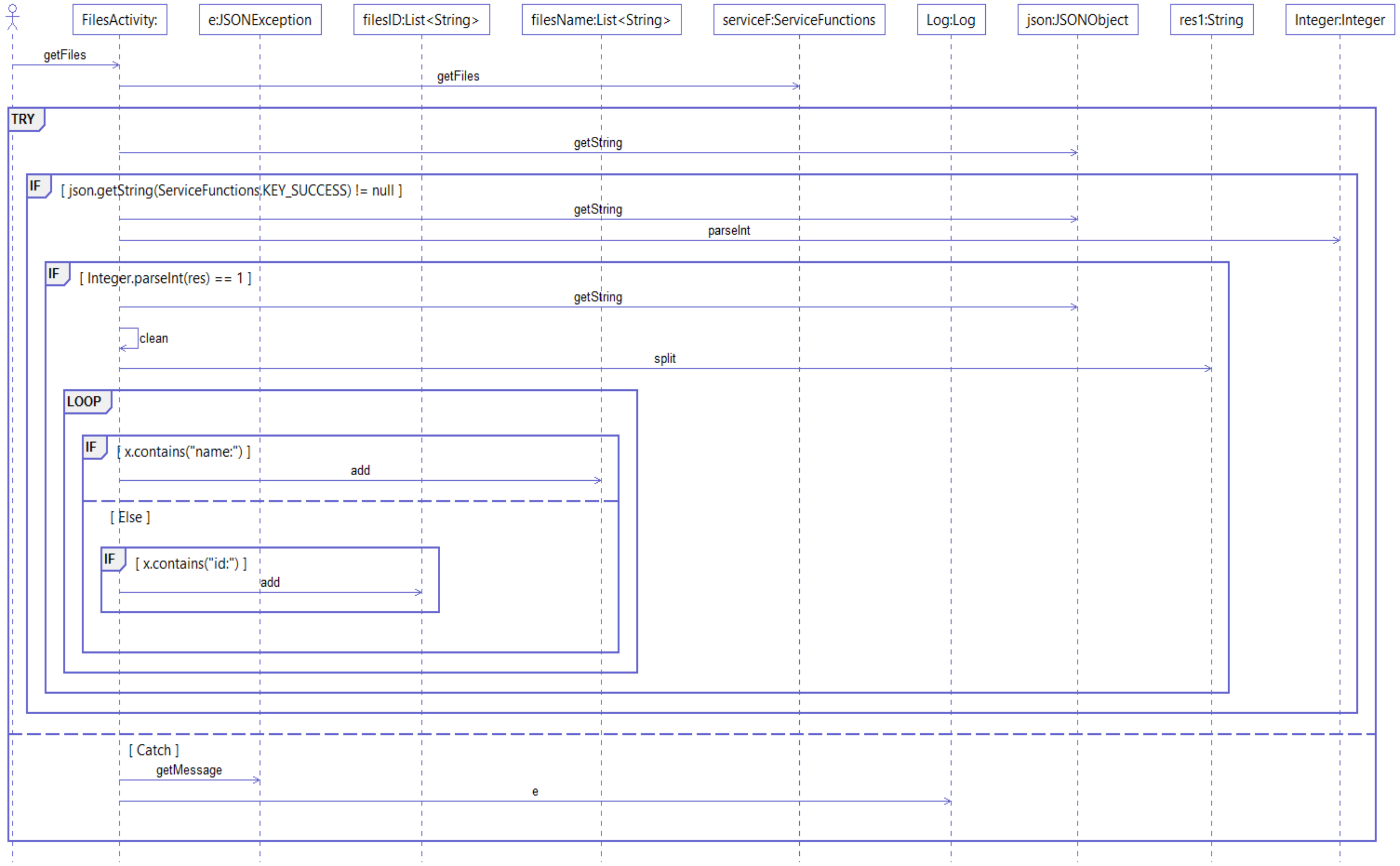


AnnouncementsActivity:
e:JSONException
announcementText:List<String>
serviceF:ServiceFunctions
announcementID:List<String>
announcementDate:List<String>
Log:Log
json:JSONObject
res1:String
announcementName:List<String>
Integer:Integer
getAnnouncements
getAnnouncements
TRY
getString
IF [json.getString(ServiceFunctions.KEY_SUCCESS) != null]
getString
parseInt
IF [Integer.parseInt(res) == 1]
getString
clean
split
LOOP
IF [x.contains("name:")]
add
[Else]
IF [x.contains("timestamp:")]
add
[Else]
IF [x.contains("text:")]
add
[Else]
IF [x.contains("id:") && !x.contains("cid:")]
add
[Catch]
getMessage
e

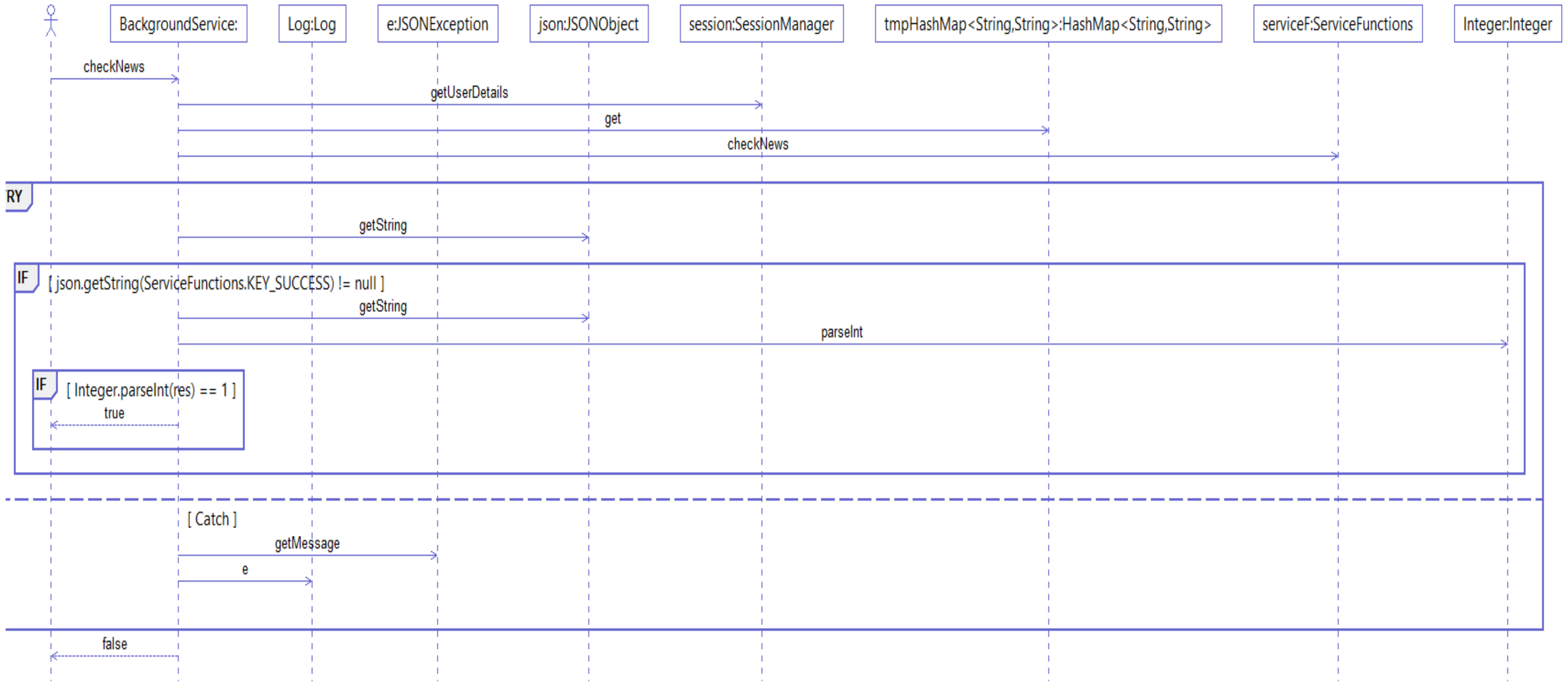
Appendix B.3 : Retrieve User's Announcements List Sequence Diagram



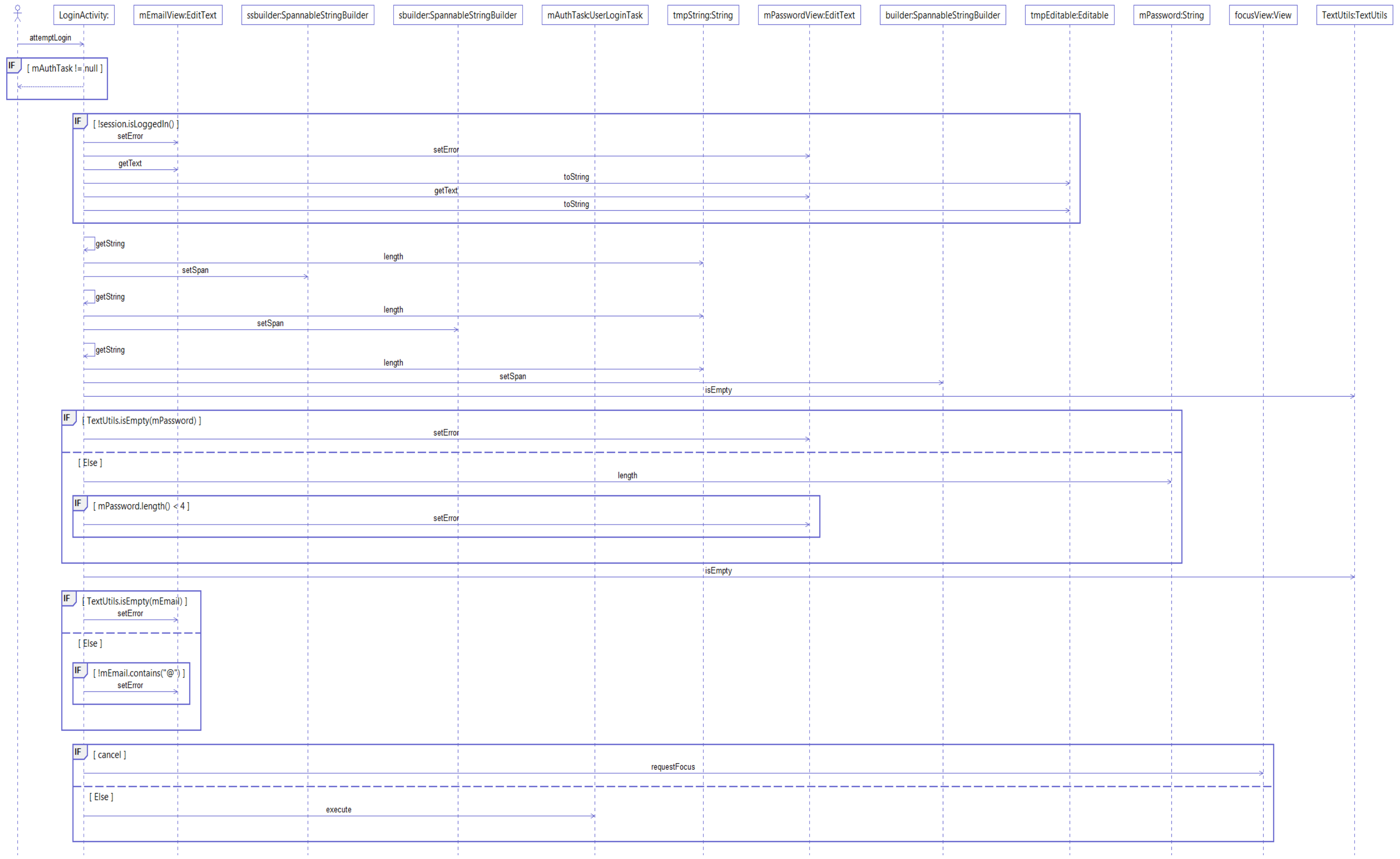
Appendix B. 4: Create New Announcement Sequence Diagram



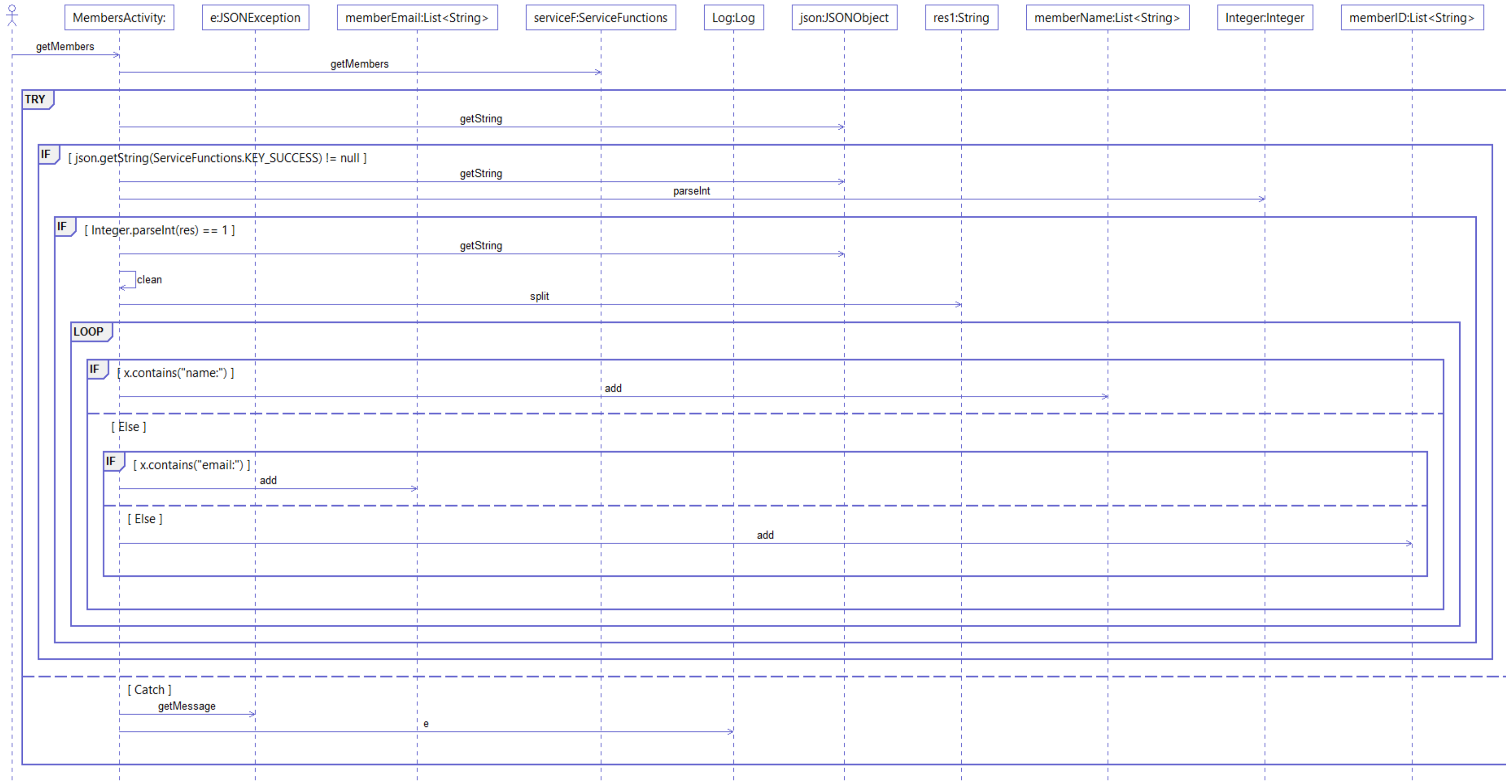
Appendix B.5: Retrieve Course's Files List Sequence Diagram



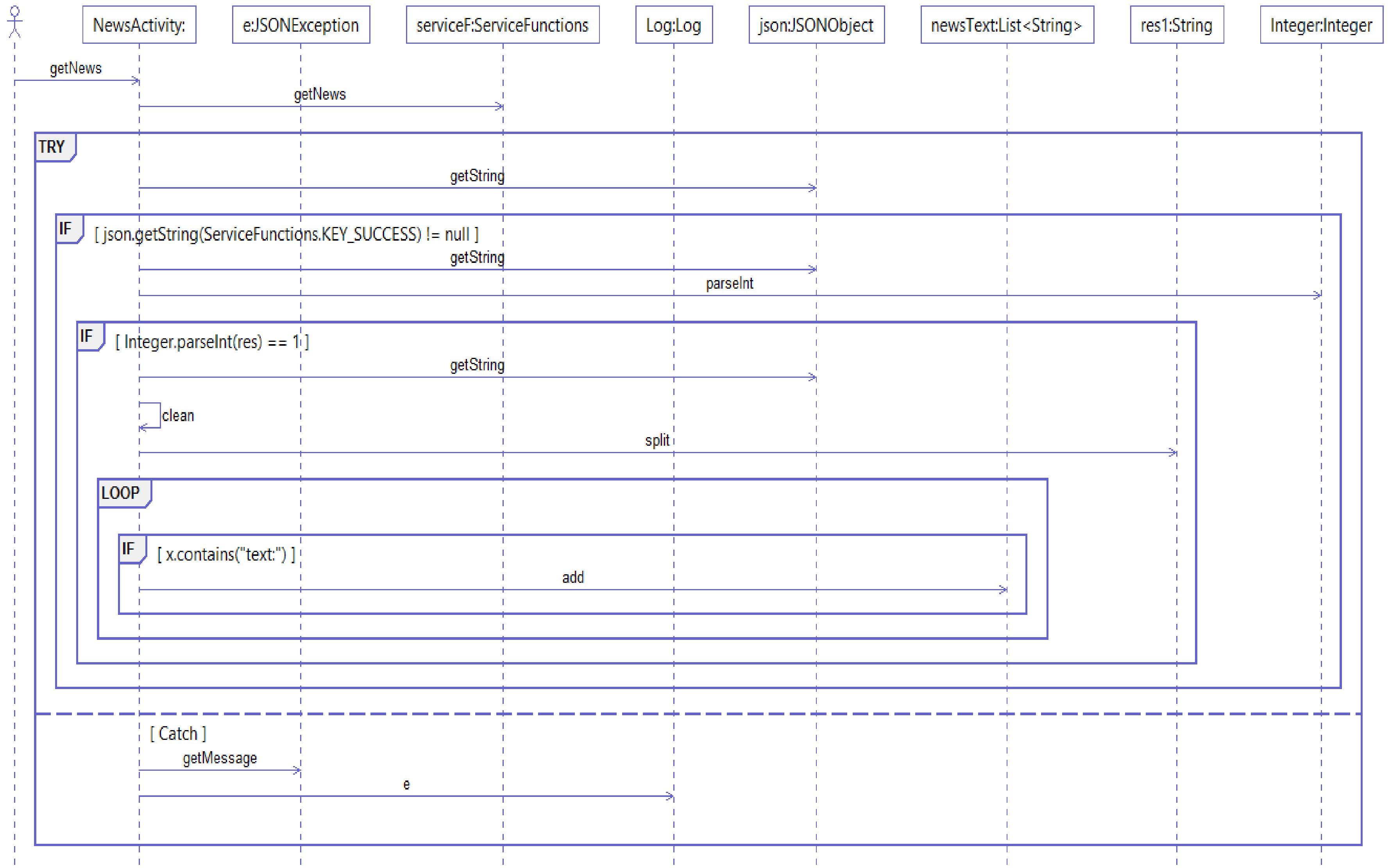
Appendix B. 6: BackGround Service Sequence Diagram



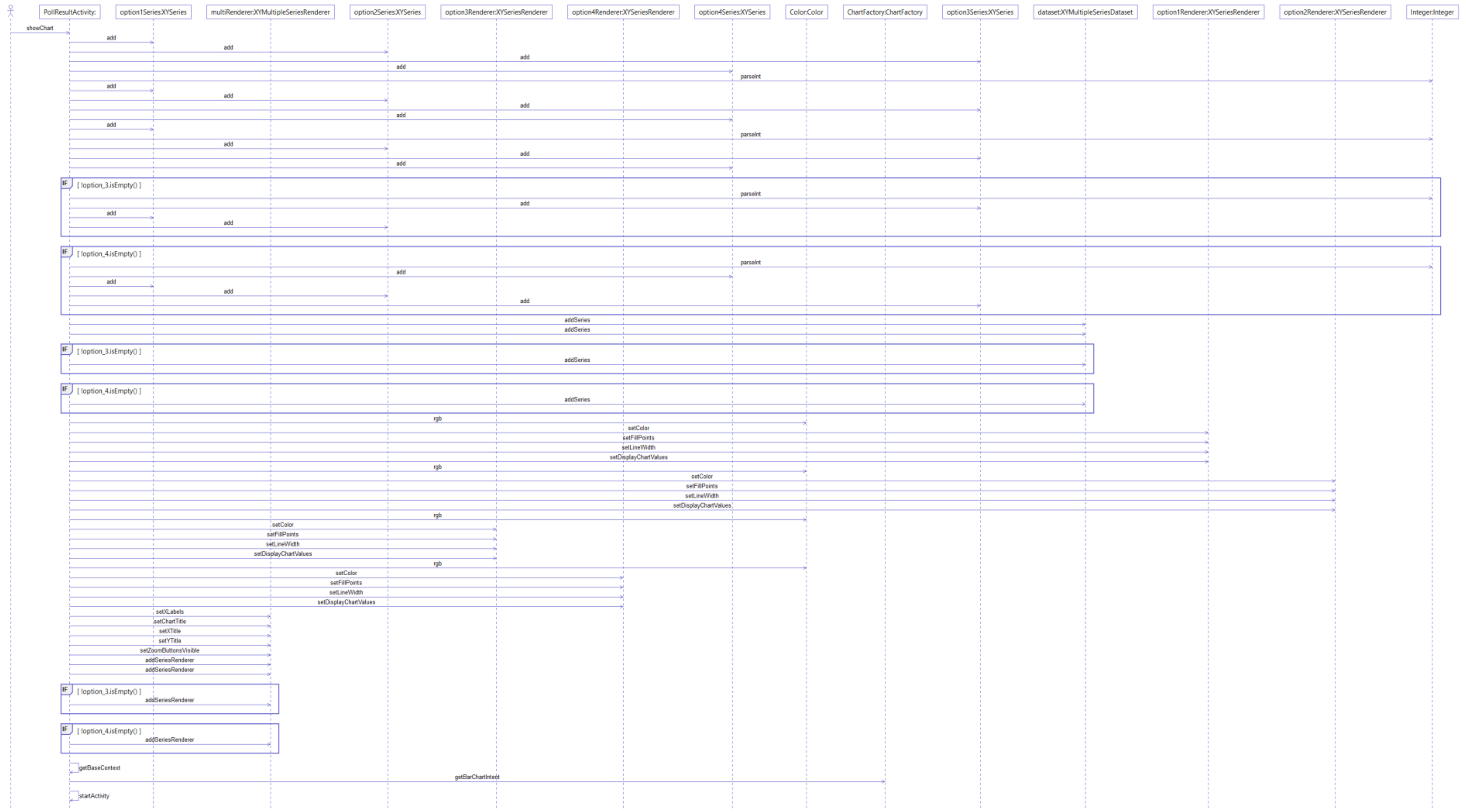
Appendix B. 7: Attempt Login Sequence Diagram



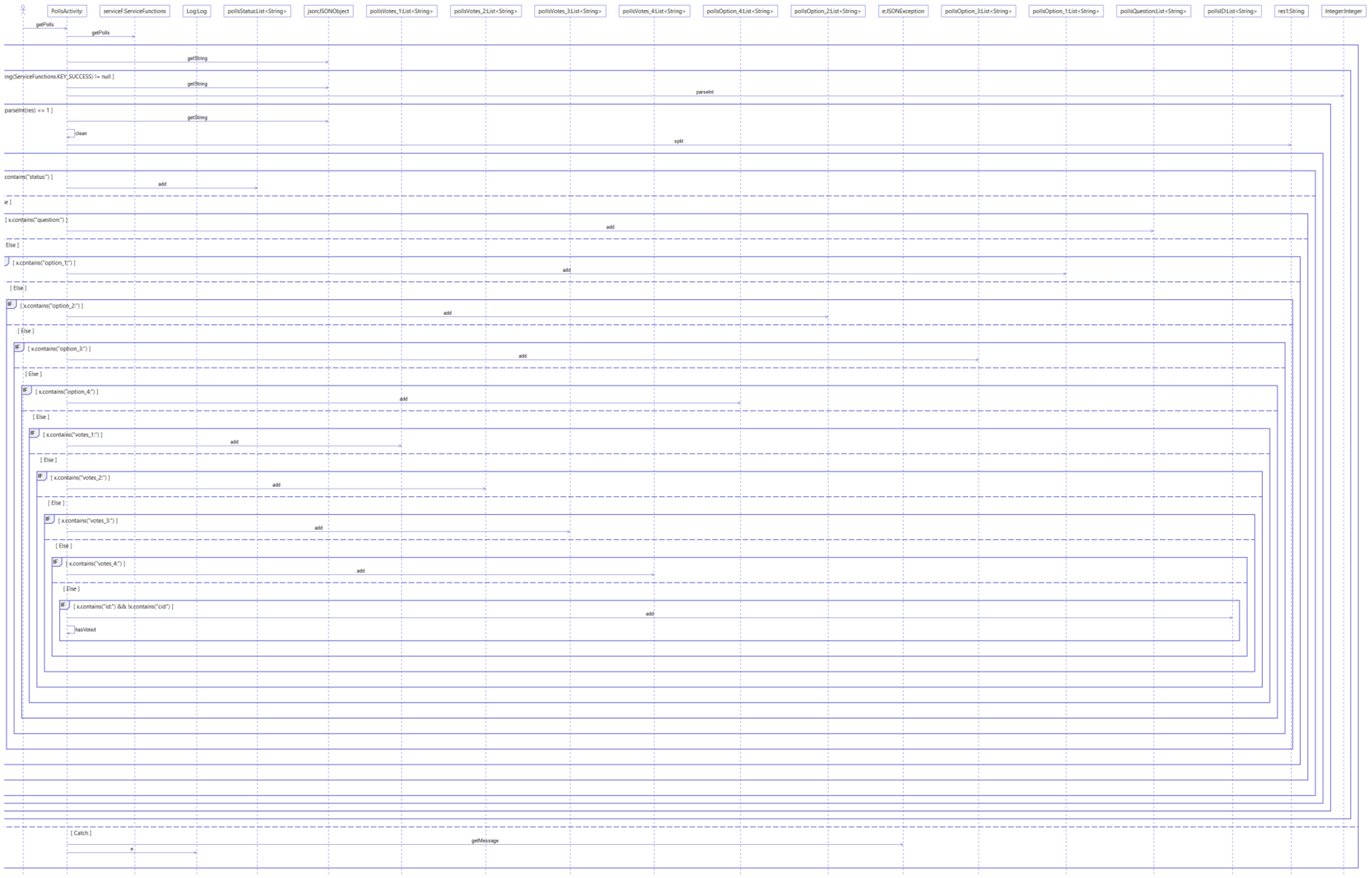
Appendix B.8: Retrieve Course's Members List Sequence Diagram



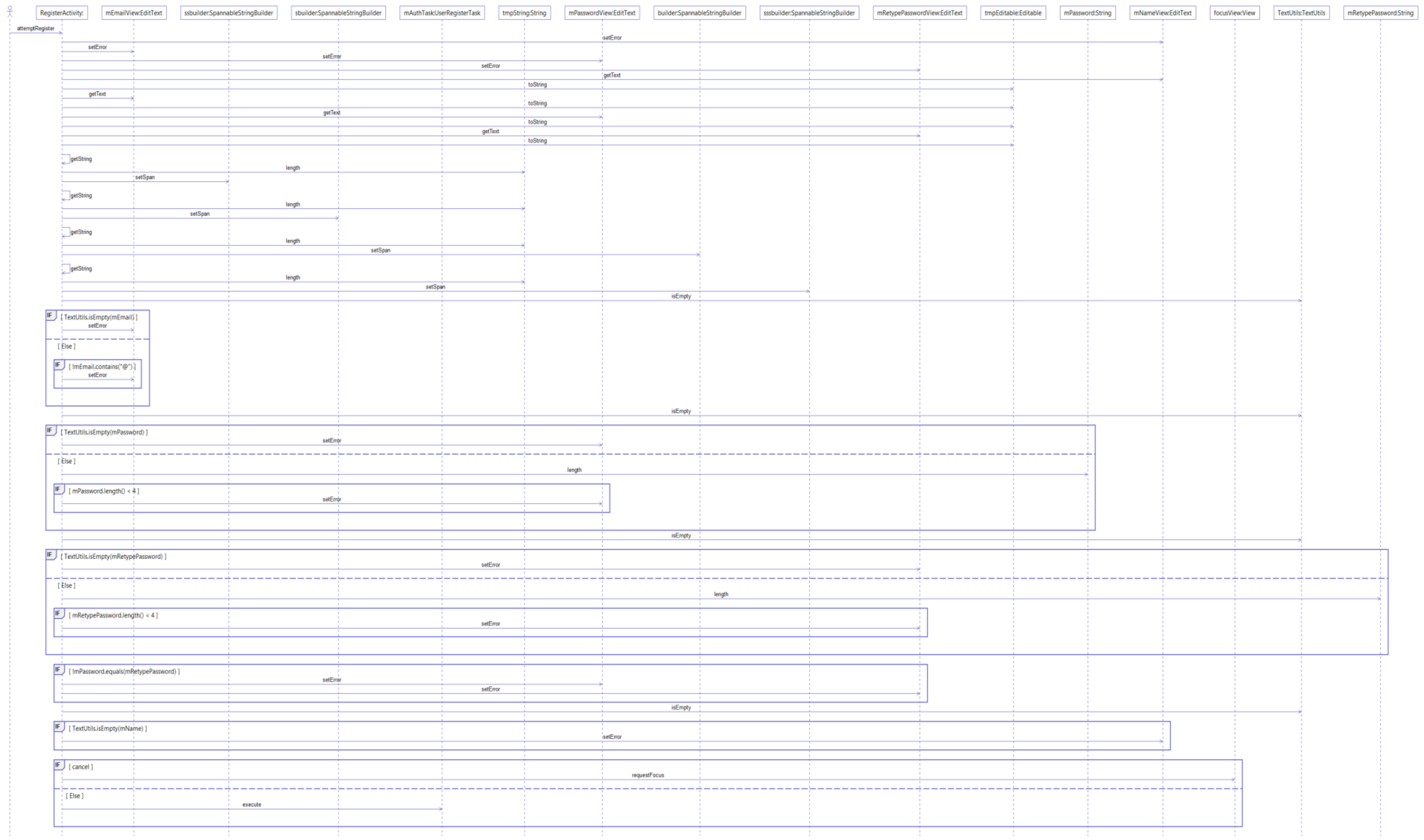
Appendix B.9: Retrieve User's News



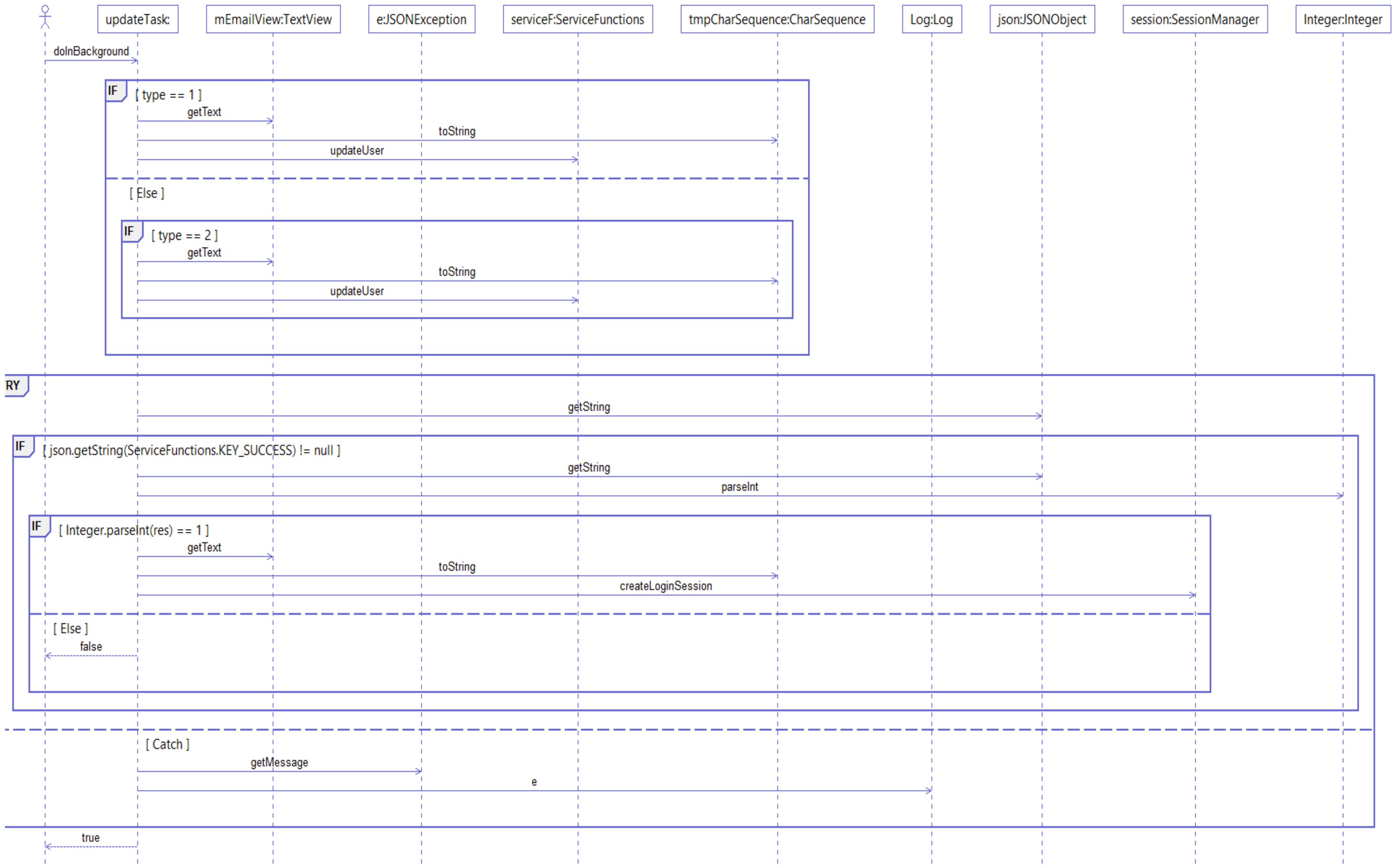
Appendix B.10: Show Poll's Result Chart Sequence Diagram


PollsActivity:
serviceF:ServiceFunctions
Log:Log
pollsStatus:List<String>
json:JSONObject
pollsVotes_1:List<String>
pollsVotes_2:List<String>
pollsVotes_3:List<String>
pollsVotes_4:List<String>
pollsOption_4:List<String>
pollsOption_2:List<String>
e:JSONException
pollsOption_3:List<String>
pollsOption_1:List<String>
pollsQuestion:List<String>
pollsID:List<String>
res1:String
Integer:Integer
getPolls
getPolls
getString
getString
parseInt
getString
clean
split
add
hasVoted
getMessage
e
[Else]
[Catch]

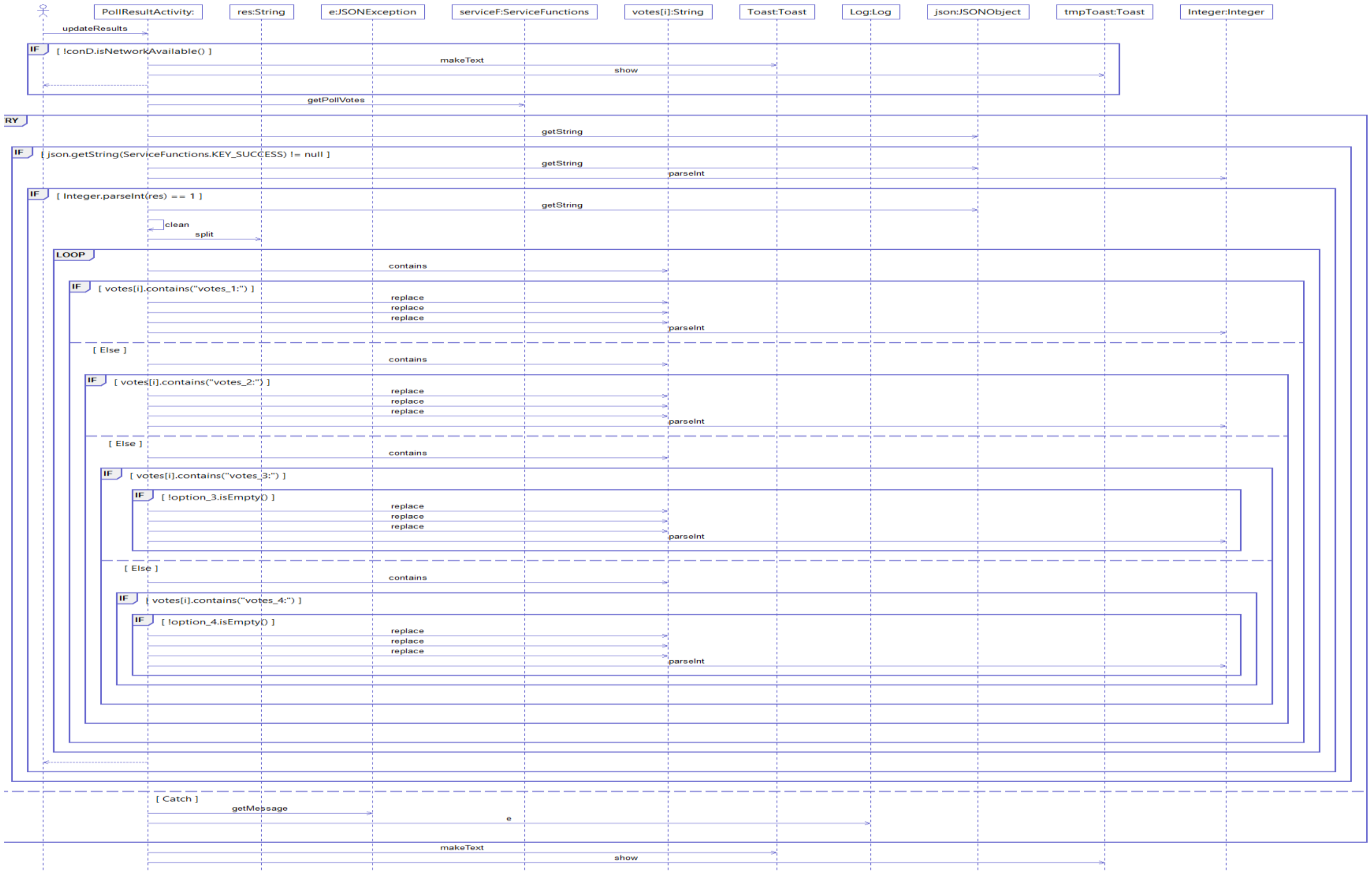
Appendix B.11: Retrieve Course's Polls List Sequence Diagram


RegisterActivity:
mEmailView:EditText
ssbuilder:SpannableStringBuilder
sbuilder:SpannableStringBuilder
mAuthTask:UserRegisterTask
tmpString:String
mPasswordView:EditText
builder:SpannableStringBuilder
sssbuilder:SpannableStringBuilder
mRetypePasswordView:EditText
tmpEditable:Editable
mPassword:String
mNameView:EditText
focusView:View
TextUtils:TextUtils
mRetypePassword:String
attemptRegister
setError
getText
toString
getString
length
setSpan
isEmpty
[TextUtils.isEmpty(mEmail)]
[Else]
[!mEmail.contains("@")]
[TextUtils.isEmpty(mPassword)]
[mPassword.length() < 4]
[TextUtils.isEmpty(mRetypePassword)]
[mRetypePassword.length() < 4]
[!mPassword.equals(mRetypePassword)]
[TextUtils.isEmpty(mName)]
[cancel]
requestFocus
execute

Appendix B. 12: Attempt Register Sequence Diagram

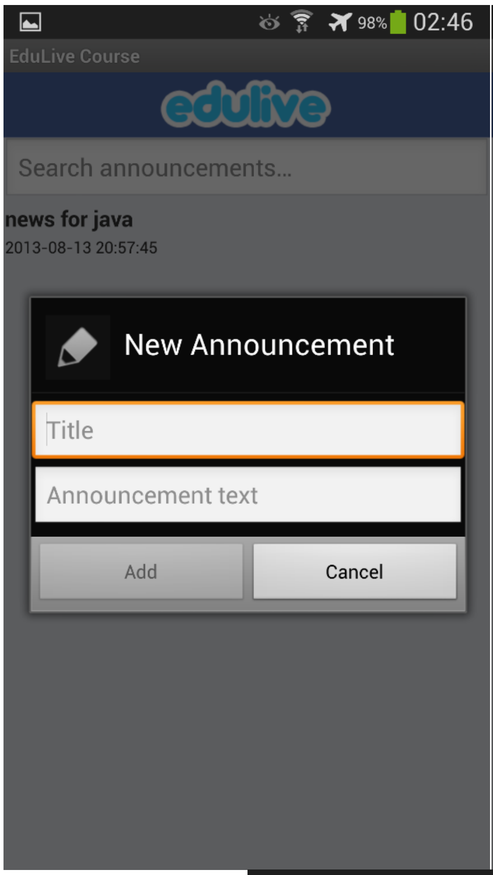


Appendix B.13: Update User's Details Sequence Diagram

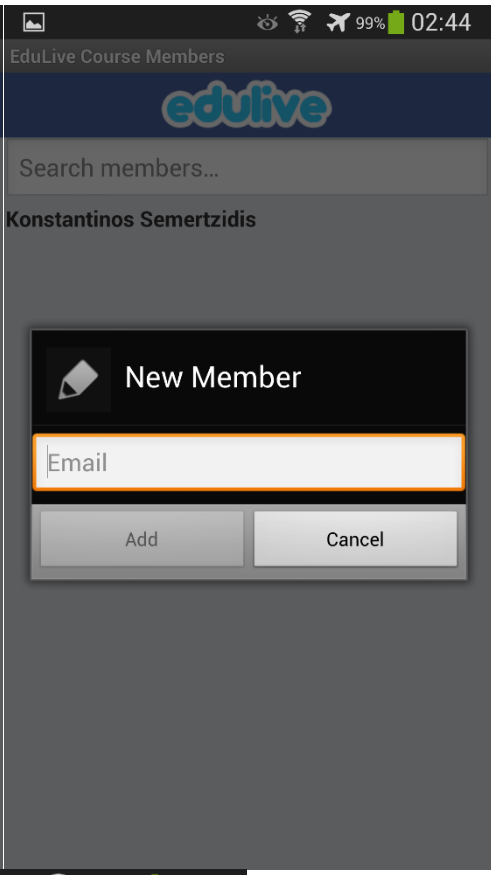


Appendix B.14: Update Poll's Results Sequence Diagram

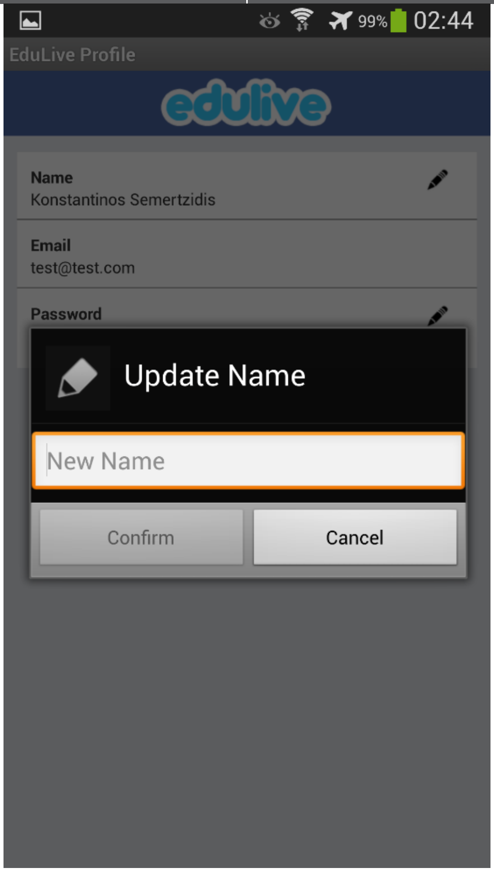
User Interface Figures



news for java
2013-08-13 20:57:45



Konstantinos Semertzidis



Name
Konstantinos Semertzidis
Email
test@test.com
Password

Database Tables' Attributes

courses_files		
Attribute	Type	Description
id	INT	The unique id of the file.
cid	INT	The id of the course.
name	VARCHAR	The name of the file.
courses_announcements		
Attribute	Type	Description
id	INT	The unique id of the announcement.
cid	INT	The id of the course.
name	VARCHAR	The name of the announcement.
text	VARCHAR	The text of the announcement.
timestamp	TIMESTAMP	The time when the announcement created.
courses_polls		
Attribute	Type	Description
id	INT	The unique id of the poll.
cid	INT	The id of the course.
status	TINYINT	The status open or closed (0-1) of the poll.
question	VARCHAR	The question of the poll.
option_1	VARCHAR	The option 1 of the poll.
option_2	VARCHAR	The option 2 of the poll.
option_3	VARCHAR	The option 3 of the poll.
option_4	VARCHAR	The option 4 of the poll.
votes_1	SMALLINT	The votes of option 1 of the poll.
votes_2	SMALLINT	The votes of option 2 of the poll.
votes_3	SMALLINT	The votes of option 3 of the poll.
votes_4	SMALLINT	The votes of option 4 of the poll.
users_polls		
Attribute	Type	Description
uid	INT	The unique id of the user.
pid	INT	The unique id of the poll.
users_courses		
Attribute	Type	Description
uid	INT	The unique id of user.
cid	INT	The unique id of the course.
users_news		
Attribute	Type	Description
id	INT	The unique id of news.
uid	INT	The user id of the news.
text	VARCHAR	The text of the news.
timestamp	TIMESTAMP	The time when news created.
isRead	INT	The status is read or not of the news.

Appendix C

Application's Test Cases

Test Reference	TS-07
Tested Requirement	FR03
Test Content	Checks that the application allows user to log out at any time pressing the log out button in the application's menu.
Input	User presses on device's menu button, then presses on the displayed logout button in all application's screens and then presses on confirm button.
Pass Criteria	Application shows the menu in all screens, and when user presses on the logout button, it shows a dialog with confirm and cancel button. When user presses on confirm button the application redirects the user to the log in screen.

Test Reference	TS-08
Tested Requirement	FR04/06
Test Content	Checks that the application provides error recognition messages during the sign in, registration, detail changing process.
Input	User fills out the login form with an email that doesn't exist. User fills out the registration form without adding anything in any form. User fills the update form without retyping the password.
Pass Criteria	Application shows error recognition message in the forms, showing the error message.

Test Reference	TS-09
Tested Requirement	FR07
Test Content	Checks that the application allows user to create a new course.
Input	User presses on menu button and presses on new course. Then, it fills out the displayed dialog and press confirms.
Pass Criteria	Application shows the menu buttons. Then, it shows the dialog for creating the course and finally shows the message that the course has been created.

Test Reference	TS-10
Tested Requirement	FR09
Test Content	Checks that the application allows course's owner to add other users in the course.
Input	Users adds an email to a form and presses on add button.
Pass Criteria	Application shows the form and when user presses on the add button shows that the user has been added. Then the user is added in the members list of the course.

Test Reference	TS-11
Tested Requirement	FR10

Test Content	Checks that the application allows user to remove a course from his list.
Input	User presses on remove course in the context menu.
Pass Criteria	If user is the course owner it removes the course from the user's list and from all members. If user is a member it removes the course only from the user's list.

Test Reference	TS-12
Tested Requirement	FR11
Test Content	Checks that the application allows a course's owner to edit the course.
Input	User adds the new details for the course (such as name, department, institution)
Pass Criteria	Application shows the message that the course has been edited and change the details in the courses list and in all course's members.

Test Reference	TS-13
Tested Requirement	FR12
Test Content	Checks that the application allows the course's owner to add an announcement.
Input	User adds the details for the announcement (such as title, text).
Pass Criteria	Application displays the new announcement form, and then shows the announcement in all members course's announcements list.

Test Reference	TS-14
Tested Requirement	FR13
Test Content	Checks that the application allows a course's owner to add files.
Input	Users navigating the file manager and chooses a file.
Pass Criteria	Application shows the file manager and when user chooses the file it starts the uploading process and in parallel shows a progress dialog. When the operation is done the file is available in all members of the course.

Test Reference	TS-15
Tested Requirement	FR14
Test Content	Checks that the application displays the course's announcement in a list when user is navigating in announcements screen.
Input	The user presses on announcements button.
Pass Criteria	Application displays a progress dialog and then displays the announcements in the list.

Test Reference	TS-16
Tested Requirement	FR18
Test Content	Checks that the application allows course's owner to lock/unlock a poll.
Input	The users presses on lock/unlock item on context menu.

Pass Criteria	Application displays the context menu and then shows the message of locking/unlocking of the poll. After the lock operation nobody of the members can open the poll.

Test Reference	TS-17
Tested Requirement	FR16
Test Content	Checks that the application allows user to create a new poll.
Input	User presses on menu button and presses on new poll. Then, it fills out the options form and the question form and presses on submit button.
Pass Criteria	Application shows the menu buttons. Then, it shows the screen for creating the poll and finally shows the message that the poll has been created.

Test Reference	TS-18
Tested Requirement	FR21
Test Content	Checks that the application allows the course's members to view the results of a poll.
Input	User presses on results item of the context menu.
Pass Criteria	Applications displays the context menu and after the pressed of button it redirects to the results screen of the poll.

Test Reference	TS-19
Tested Requirement	FR19
Test Content	Checks that the application displays the course's polls in a list when user is navigating in polls screen.
Input	The user presses on polls button.
Pass Criteria	Application displays a progress dialog and then displays the polls in the list.

Test Reference	TS-20
Tested Requirement	FR22
Test Content	Checks if application doesn't allow users to open a poll if they have already voted.
Input	User presses on open item of context menu.
Pass Criteria	Application shows a message that user has already voted and doesn't redirect him to the poll screen.

Test Reference	TS-21
Tested Requirement	FR23
Test Content	Checks that the application allows a course's owner to delete an announcement.
Input	User presses on delete item of context menu.
Pass Criteria	Application shows a message that the announcement has been deleted and deletes it from all course's members.

Test Reference	TS-22

Tested Requirement	FR24
Test Content	Checks that the application allows a course’s owner to edit the announcement.
Input	User adds the new details for the announcement(such as title, text)
Pass Criteria	Application shows the message that the announcements has been edited and change the details in the announcements list and in all course’s members.

Test Reference	TS-23
Tested Requirement	FR25
Test Content	Checks that the application allows a course’s owner to delete a file
Input	User presses on delete item of context menu.
Pass Criteria	Application shows a message that the file has been deleted and deletes it from all course’s members.

Test Reference	TS-24
Tested Requirement	FR27
Test Content	Checks that the application informs user that has been added to a new course.
Input	Nothing, it runs on background.
Pass Criteria	If user has been added to a course, the application shows a notification message in the device bar. When user presses on it, the application opens the news screen.

Test Reference	TS-25
Tested Requirement	FR17
Test Content	Checks that the application allows a course’s owner to delete a poll.
Input	User presses on delete item of context menu.
Pass Criteria	Application shows a message that the poll has been deleted and deletes from all course’s members.

Test Reference	TS-26
Tested Requirement	FR26
Test Content	Checks that the system provides recognition messages to the user for any action that is available in each course.
Input	Users navigates through all screens after pressing a course from the course’s list. Then, he uses all the available options to create, edit, and delete items (such as announcements, files, members, polls).
Pass Criteria	Application shows recognition messages for every user’s action.