الجامـــعة الافتراضيّة السوريّة
ماجستير التأهيل والتخصص في علوم الحاسوب

تطبيق المساعد الجامعي الشخصي

مشروع أعد لنيل درجة الماجستير في علوم الحاسوب

إعداد الطالب

علاء عيسى علي

بإشراف

الدكتور عصام سلمان

العام الدراسي 2025

Syrian Virtual University

Master's Degree in Computer Science

Personal University Assistant App

A project prepared to obtain a Master's degree in Computer Science

Prepared by

Alaa Issa Ali

Supervised by

Dr. Essam Salman

Year 2025

الاهداء والشكر

أحمد الله وأشكره وبه استعين، وادعوه دعوةً صادقةً الى كل من كان لي سنداً ومعلماً وزميلاً وصديقاً ومساهماً فيما وصلت له.

وأخص بالشكر:

من كان لعلمه وتوجيهاته وسعة صدره وتعاونه الأثر الكبير في انجاز هذا المشروع (أستاذي ومشرفي العزيز د. عصام سلمان)

الكادر التعليمي والإداري لبرنامج ماجستير علوم الحاسوب وعلى رأسهم مديرة البرنامج الدكتورة سيرا أستور على كل ما قدموه من علم وارشاد ودعم متواصل.

أصدقائي وزملائي الذين شاركتهم هذه الرحلة التعليمية بكل ما فيها من تحديات وانجازات، كانت رحلة مميزة بفضل وجودكم.

أهدي هذا العمل الى الأغلى على قلبي، الذين كانوا السند الدائم وقدموا التضحيات لأكون دوما في أفضل حال، أدامكم الله بالصحة والعافية .... (أمي وأبي الغاليين)

فهرس المحتويات

فهرس الجداول

فهرس الأشكال

الملخص

تمثل إدارة الحياة الأكاديمية للطلاب في الجامعة الافتراضية السورية تحديًا كبيرًا بسبب تشتت الخدمات بين ثلاث منصات منفصلة هي نظام المعلومات SVUIS، نظام إدارة التعلم LMS، نظام البريد الإلكتروني SVUE، مما يؤدي إلى صعوبة في المتابعة وعدم كفاءة في الوصول إلى الخدمات والموارد التعليمية.

انطلاقًا من هذه المشكلة، يقدم هذا المشروع تصميم وتنفيذ تطبيق جوال متكامل باسم "Waseet" يعمل على نظام أندرويد، بهدف توحيد الخدمات الأكاديمية والموارد التعليمية في منصة واحدة. يعتمد التطبيق على منهجية النموذج الأولي التطوري (Evolutionary Prototyping) لضمان مواءمة المتطلبات وتطويرها بشكل تكراري بناءً على ملاحظات المستخدمين.

يتميز التطبيق بمجموعة من الميزات المتكاملة، أهمها: (1) دمج الخدمات الأكاديمية الأساسية (الجداول، النتائج، المحتوى التعليمي) (2) نظام إشعارات ذكي للتنبيه بالمحاضرات والمواعيد الهامة (3) موارد تعليمية إضافية يضيفها مرشدون متطوعون (4) روبوت دردشة ذكي للمساعدة في الإجابة على الأسئلة (5) أدوات مساعدة مثل المذكرة والتقويم الأكاديمي.

النتيجة النهائية هي منتج كامل يعمل كمنصة تعليمية شاملة، يستفيد منه جميع الطلاب حيث يمنحهم تجربة موحدة وسلسة، ويمكن الإدارة من مراقبة العمليات وإرسال الإشعارات والأخبار. يُوصى بالتطبيق كنموذج قابل للتطوير يمكن تطبيقه في مؤسسات تعليمية أخرى

.

الكلمات المفتاحية: نظام إدارة الطلاب, واجهات المستخدم و تجربة المستخدم, تطبيق الجوال, المحادثة الذكية, إدارة الملاحظات, إدارة التنبيهات.

Abstract

Managing academic life for students at the Syrian Virtual University poses a significant challenge due to the fragmentation of services across three separate platforms (SVUIS, LMS, and SVUE email system), leading to difficulties in tracking and inefficient access to educational services and resources.

To solve this issue, this project propounds the design and development of an integrated mobile application dubbed "Waseet" for the Android system with the intention of aggregating academic services and educational material in a centralized system. The application embraces an Evolutionary Prototyping approach to achieve requirements consistency and incremental development with the feedback of the users.

The application was developed using modern technologies, including the Flutter framework for the frontend, Django for the backend and APIs, SQLite for local storage, and Firebase Cloud Messaging for notifications.

The application features a set of integrated functionalities, most notably: (1) integration of core academic services (schedules, results, educational content) (2) an intelligent notification system for lectures and important deadlines (3) additional educational resources added by volunteer mentors (4) an AI-powered chatbot for answering questions (5) helper tools such as notes and an academic calendar.

The outcome is a complete product that functions as a comprehensive educational platform, benefiting all students by providing a unified and seamless experience, while enabling administrators to monitor operations and send notifications and news. The application is recommended as a scalable model applicable to other educational institutions.

Keywords: Student Management System, User Interfaces and User Experience, Mobile App, Smart Chat, Notes Management, Alerts Management.

الفصل الأول: الإطار النظري العام

1.1- مقدمة نظرية:

1.1.1- مقدمة:

في ظل التطور التكنولوجي المتسارع، أصبحت الهواتف الذكية جزءًا لا يتجزأ من الحياة اليومية، لا سيما في المجال التعليمي مما يبرز الحاجة إلى تطبيقات تعليمية متكاملة تُسهِّل إدارة الحياة الأكاديمية وحسب مقالة عربية **(لماذا تعد تجربة الهاتف المحمول الجيدة أمرًا أساسيًا لتحفيز تاريخ 24:سبتمبر 2019)** تعمل تطبيقات الهاتف المحمول على تحسين تجربة المستخدم لأنظمة إدارة الطلاب من خلال توفير وصول سهل إلى المعلومات الأكاديمية وأدوات الاتصال وموارد التعلم على الهواتف الذكية والأجهزة اللوحية.

عرفت "زينب الشربيني"(2012 25) تكنولوجيا التعليم النقال بأنها: " توظيف الأجهزة الرقمية اللاسلكية الصغيرة للقيام بوظائف تعليمية عديدة، مثل الاتصالات الصوتية، وخدمات ارسال واستقبال وعرض الرسائل النصية القصيرة، والبريد الإلكتروني، وتصفح الويب، وتصميم المحتوى الإلكتروني ونشره ".

لذلك بالنسبة لاي طالب جامعي فإن وجود تطبيق موحد يُتيح الوصول إلى المعلومات الأكاديمية (كالجدول الدراسي، الدرجات، والمواد التعليمية) ويُقدّم خدمات اخرى مثل التنبيهات وتنظيم المهام، سيكون امرا ذو أهمية كبيرة.

بالإضافة إلى ذلك، يُعتبر الجوال وسيطًا مثاليًا لمرافقة الطالب داخل الحرم الجامعي وخارجه، مما يجعله منصة فعّالة لتقديم خدمات سريعة ومُباشرة.

وحاليا نجد إن التعلم عبر الأجهزة المحمولة يعزز من تجربة الطالب التعليمية، ويُسهم في تسهيل الوصول إلى الموارد الدراسية، مما يجعل من تطوير تطبيقات ذكية تعليمية حاجة ضرورية في عصرنا الرقمي.

وبالنظر لما سبق فإن تطوير تطبيق جامعي ذكي يدعم الإشعارات التلقائية ويُوفّر معلومات دقيقة في الوقت الفعلي ومصادر مساعدة للتعلم أصبح ليس فقط حاجة ملحة، بل أداة استراتيجية لتحسين جودة التجربة التعليمية في العصر الرقمي.

2.1.1- تطبيق الهاتف المحمول (Mobile App):

تطبيق المحمول هو برنامج كمبيوتر مصمم ليعمل على الهواتف الذكية، وأجهزة الكمبيوتر اللوحي وغيرها من الأجهزة النقالة.

يتم تنزيل هذه التطبيقات عموما من منصات توزيع التطبيقات وأشهرها (iOS Store App) و (Google Play store).

3.1.1- نظام إدارة الطلاب (Student Management System):

هو منصة إلكترونية متكاملة تهدف إلى إدارة وتنظيم العمليات الأكاديمية والإدارية المتعلقة بالطلاب في المؤسسات التعليمية. يشمل هذا النظام مجموعة من الأدوات والتطبيقات التي تساعد في تسجيل الطلاب، إدارة الجداول الدراسية، متابعة الحضور، وتخزين السجلات الأكاديمية. كما يوفر هذا النظام واجهة مستخدم سهلة الاستخدام وقاعدة بيانات مركزية تضمن تخزين البيانات بشكل آمن ومنظم، مما يتيح الوصول إليها بسهولة عند الحاجة.

4.1.1- واجهة المستخدم (UI):

إن واجهة المستخدم في مجال التصميم الصناعي لتفاعل الإنسان مع الآلة، هي مجموعة من الوسائل التي يتفاعل بها الأشخاص (المستخدمون) مع منتج أو خدمة رقمية كاستخدام آلة، أو جهاز، أو برنامج حاسوبي، أو أي أداة معقدة أخرى (النظام). ويتم الوصول إلى هذه الوسائل بعدة طرق مثل الأزرار والرموز.

5.1.1- تجربة المستخدم (UX):

تجربة المستخدم بالإنجليزية(User experience): يُطلق عليها اختصاراً UX ، هي كل ما يرتبط بسلوك وموقف وإحساس المستخدم حيال استخدامه منتجاً أو نظاماً أو خدمةً معيّنة.

تُبْرِز تجربة المستخدم الجوانب القيمة والعاطفية والتجريبية وذات المعنى في التفاعل بين الإنسان والحاسب وملكية المنتج، ولكن تتضمن أيضاً تصورات أي شخص حول الجوانب العملية مثل الفائدة وسهولة الاستخدام وكفاءة النظام.

تعتبر تجربة المستخدم شخصيةً في الطبيعة، لأنها تكون عن مشاعر الشخص وأفكاره عن النظام. تعد تجربة المستخدم ديناميكية، لأنها تتغير مع الوقت عندما تتغير الظروف.

2.1- مشكلة البحث ومبررات المشروع:

1.2.1- مشكلة البحث:

يواجه طلاب الجامعة الافتراضية السورية (SVU) تحديات كبيرة في إدارة حياتهم الأكاديمية بسبب:

- **التشتت الرقمي:**

حاجة الطلاب للتنقل بين ثلاث منصات رئيسية بشكل منفصل:

- نظام معلومات وامتحانات الجامعة (SVUIS)
- نظام إدارة التعلم (LMS)
- نظام البريد الإلكتروني (SVUE)

- **صعوبة المتابعة:**

- عدم وجود نظام موحد للتنبيهات بخصوص الاحداث الهامة كالمحاضرات القادمة مثلاً
- عدم تأكيد وصول إشعارات الامتحانات والواجبات
- صعوبة تتبع المواعيد الهامة

2.2.1- مبررات المشروع:

يأتي هذا المشروع كحل شامل لهذه التحديات بناءً على المبررات التالية:

- **مبررات متعلقة بالكفاءة:**

- **توفير الوقت:** دمج ثلاث منصات رئيسية في مكان واحد وسهولة التنقل فيما بينها من خلال واجهات التطبيق
- **الوصول السريع:** إمكانية الوصول للمحاضرة القادمة وتشغيل التنبيه بنقرة واحدة
- **التنظيم الأمثل:** نظام موحد للإشعارات يشمل:
    - تنبيهات المحاضرات
    - تنبيهات الامتحانات
    - مواعيد تسليم الواجبات

- **مبررات تعليمية:**

- **دعم التعلم الذاتي:** من خلال نظام الاقتراحات التعليمية الذي يوفر:
    - روابط لمصادر إضافية مثل الفيديوهات التعليمية والشروحات والمقالات المفيدة
    - ملفات مساعدة مثل الكتب الالكترونية
- **تحسين الأداء الأكاديمي:** عبر أدوات مدمجة مثل:
    - حاسبة المعدل التراكمي
    - التقويم الأكاديمي
    - نظام إدارة الملاحظات

- **مبررات تقنية:**

- **تجربة مستخدم محسنة:** واجهات حديثة بتصميم أنيق تسهل الوصول للموارد المطلوبة
- **تكامل الأنظمة:** ربط الأنظمة الثلاثة (SVUIS، LMS، SVUE) في تطبيق واحد
- **الأمان والخصوصية:** حماية بيانات الطلاب ضمن بيئة موحدة

- **مبررات عملية:**

- **حل مشاكل حقيقية:** بناءً على اقتراحات الطلاب وتغطية نقاط الضعف الحالية

- **توفير الجهد:** إلغاء الحاجة لتطبيقات متعددة
- **التكيف مع نظام:** تصميم مخصص لنظام التعليم الافتراضي في الجامعة الافتراضية السورية

3.2.1- القيمة المضافة للمشروع:

- **الابتكار في التكامل:** أول تطبيق يدمج الأنظمة الثلاثة لـ SVU
- **الذكاء في التنبيهات:** نظام إشعارات يراعي أولويات الطالب
- **الدعم التعليمي الشامل:** ليس فقط أدوات إدارة، بل أيضاً موارد تعليمية

4.2.1- الفروق الجوهرية عن الحلول الحالية:

مقارنة بالخدمات حالية:

- توفير حل متكامل بدلاً من خدمات منفصلة
- تصميم واجهات مخصصة لـSVU
- دعم جميع الخدمات الأكاديمية في مكان واحد
- مقارنة بالوصول عبر المتصفح:
  - سرعة أكبر في الوصول للمعلومات
  - إشعارات فورية من دون الحاجة الى استخدام المتصفح
  - تجربة مستخدم أكثر سلاسة

5.2.1- التأثير المتوقع:

- **تحسين الأداء الأكاديمي:** بحكم تسهيل التجربة التعليمية فمن المتوقع تحسن الأداء الأكاديمي للطلاب بنسبة جيدة.
- **توفير الوقت:** تقليل الوقت الضائع في التنقل بين الأنظمة بنسبة جيدة عبر توحيد الوصول من مكان واحد
- **رفع مستوى الرضا:** تحسين تجربة الطالب مع النظام التعليمي لـSVU
- **نموذج قابل للتطوير:** يمكن تطبيقه على جامعات أخرى

6.2.1- الجهات المستفيدة:

المشروع موجه الى الطلاب في المرحلة الجامعية أو الماجستير وبشكل خاص قمنا بتخصيص التطبيق ليتناسب مع جميع طلاب الجامعة الافتراضية بكافة الفروع ليكون تطبيق عملي للحالة المدروسة ويتناسب مع المتطلبات التقنية والبرمجية للجامعة.

3.1- متطلبات المشروع وقيوده:

- المتطلبات:

يمكن تلخيص متطلبات المشروع بالنقاط الآتية:

- اعتماد واجهات مستخدم بتصميم عصري ومريح، وبحيث تكون بنفس الوقت بسيطة وعملية ومفهومة وتراعي سهولة الاستخدام وتأمين جميع التفاعلات الممكنة.
- تحديد الوظائف التي سيوفرها التطبيق وتحديد حالات الاستخدام والممثلين الفاعلين فيه.
- مراعاة إمكانية التطوير وذلك بتصميم النظام بحيث يكون قابلاً للتطوير لاحقاً.
- الأمان والموثوقية من خلال استخدام التشفير وتحديد الصلاحيات بشكل واضح.
- تحقيق جميع العمليات والخدمات المطلوبة في أقسام التطبيق.
- عند الضرورة يمكن توفير تدريب للطلاب على استخدام التطبيق اما بشكل شخصي او عبر وسائل الايضاح التعليمية كتسجيل مقطع فيديو تعليمي او ملف الكتروني يشرح المميزات.

- القيود:

تتمثل قيود هذا النظام بالنقاط الآتية:

- يعمل هذا النظام كمساعد تعليمي لطلاب الجامعة الافتراضية السورية.
- سيتم الاعتماد فقط على البرمجيات مفتوحة المصدر في تطوير النظام.

- ستقتصر الواجهة الأمامية حالياً على تطبيق الجوال.
- تطبيق جوال موحد لكل المستخدمين.
- تطبيق الجوال يعمل على أجهزة الجوال الحاملة لنظام التشغيل أندرويد.
- يمكن ان تعمل بعض ميزات التطبيق بدون وجود انترنت ولكن يلزم وجود اتصال بالإنترنت لتعمل بعض الخدمات في التطبيق التي تعتمد على جلب البيانات في الزمن الفعلي.

4.1- مقارنة مع بعض التطبيقات المشابهة:

يوجد مجموعة من التطبيقات التي تقدم خدمات لطلاب الجامعات تم سابقا استخدام تطبيقين هما التطبيق الأول هو (MU student login) الخاص بالطلاب في جامعة مارواديي – الهند. (1) التطبيق الثاني هو (myUniSannio) الخاص بطلاب جامعة "Sannio" الإيطالية. (2) لذلك قمنا بإجراء مقارنة بينهما وبين التطبيق الذي نقوم بتطويره من عدة نواحي وميزات وتم تجميع النتائج في الجدول التالي:

Table 1-1- المقارنة مع تطبيقات مشابهة

تطبيق Waseet	MEFGI Student	myUniSannio	الميزة
منصة شاملة متكاملة للطلاب (أكاديمياً وتعليمياً)	بوابة دخول للطلاب للخدمات الأكاديمية	إدارة المسار الجامعي	**الهدف الأساسي**
النتائج، الجداول، المقررات، الخطة الفصلية، التخصص	النتائج، الجداول، المقررات، المنهاج	النتائج، التسجيل للامتحانات، الاستبيانات	**إدارة المسار الأكاديمي**

تنزيل محتوى ال LMS وتصفحه بدون انترنت. تصفح المحتوى الإضافي للمواد	تنزيل المحتوى التعليمي	لا يدعم	**إدارة الملفات والتعلم**
يدعم اظهار الايميلات مع وصول آني لإشعارات البريد	لا يدعم	لا يدعم	**دعم خدمة البريد الالكتروني**
يدعم تفعيل التنبيهات داخل التطبيق للمحاضرات. يدعم استقبال الاشعارات لآخر الاخبار وإشعارات المرسلة من المسؤول عن التطبيق	يدعم إشعارات المسؤول فقط	يدعم إشعارات المسؤول فقط	**التنبيهات**
تتبع حالة المدفوعات. كتابة الملاحظات. يدعم المراسلة الزكية للإجابة عن الأسئلة.	طلب إجازة / تصريح خروج من الجامعة	تتبع حالة المدفوعات	**ميزات إضافية**

تحليل نقاط القوة والتميز في تطبيق Waseet:

من خلال المقارنة أعلاه، يتضح أن تطبيق **Waseet** لا يقتصر على كونه أداة للخدمات الأكاديمية فقط، بل هو منصة تعليمية متكاملة، وهذه أبرز نقاط قوته:

1. التكامل والشمولية: بينما يركز التطبيقان الآخران على جانب محدد (إدارة المسار الجامعي أو الخدمات الأكاديمية الأساسية)، فإن **Waseet** يغطي جميع احتياجات الطالب تقريباً

في مكان واحد مثل تنبيهات المحاضرات، الملاحظات، الوصول للموارد التعليمية والبريد الإلكتروني.

2. ميزات فريدة تعزز تجربة التعلم:
   - منبه المحاضرات: ميزة عملية جداً وغير موجودة في التطبيقات المقارنة.
   - الموارد الخارجية: تحول التطبيق من مجرد عارض للمعلومات إلى أداة تعلم فعالة وإتاحة الفرصة لإضافة محتوى يثري المادة الدراسية يضيف قيمة كبيرة للتطبيق.
   - نظام الملاحظات المدمج: يمنح الطالب مساحة مخصصة داخل التطبيق للدراسة دون الحاجة للتبديل بين التطبيقات.
   - دمج نظام البريد الإلكتروني الجامعي مع إشعارات فورية تعتبر نقطة تميز كبيرة.
   - مركز الأخبار مع الإشعارات يضمن أن الطالب لن يفوت أي خبر او حدث مهم متعلق بالجامعة او بالفصل الدراسي الحالي أو بأي موضوع مهم آخر.
3. المرونة والوصول إلى المحتوى: خاصية الوصول إلى **LMS** وتحميل الملفات تجعل التطبيق بديلاً شاملاً عن المتصفح للحصول على المحتوى الدراسي، مما يوفر وقت وجهد الطالب.

وبالتالي بالمقارنة مع التطبيقات الجامعية الأخرى مثل **myUniSannio** و **MEFGI Student Login**، يبرز تطبيق **Waseet** كمنصة أكثر شمولية وتكاملاً ما يجعله ليس فقط تطبيق خدمي ولكن يمكن اعتباره بيئة تعليمية متكاملة تقدم خدمات وميزات فريدة للطالب.

5.1- التقنيات المستخدمة لإنجاز المشروع:

سيتم تنفيذ المشروع بالاعتماد على التقنيات الآتية:

1.5.1- إطار العمل (Flutter) ولغة البرمجة (Dart) لتطوير الواجهة الأمامية:

ما هو فلاتر؟

بالتعريف العام يمكن القول ان فلاتر هو إطار عمل أو حزمة أدوات تطوير البرمجيات (SDK) مفتوح المصدر طورته شركة جوجل (Google) لتطوير تطبيقات نظام الأندرويد ونظام أي أو إس ونظام ويندوز وتطبيقات الويب.

وعلى الرغم من انه يستخدم أساساً لتطوير واجهات الاستخدام بالاعتماد على صفوف معرفة مسبقا تسمى (Widgets) الا انه يتجاوز ذلك للتعامل مع العمليات البرمجية من جهة الواجهة الخلفية بالتعاون مع لغة البرمجة دارت المبني على أساسها.

يساعد إطار عمل فلاتر المصممين والمبرمجين على تشييد تطبيقات بتصميم عصري وجذاب من خلال حزم ومكتبات مبنية ومكتوبة مسبقاً وجاهزة للاستخدام ويستخدم قاعدة تعليمات برمجية واحدة مما يسهل عمليات التطوير ويوفر الوقت والجهد.

يوفر فلاتر مجموعة من الأدوات والخدمات لإنشاء الواجهة الخلفية للتطبيق، بما في ذلك التكامل مع الخدمات المستندة الى السحابة وأدوات مصادقة المستخدم وتخزين البيانات ودفع الاشعارات.

يمكن تلخيص أهم مميزات فلاتر بما يلي:

- يجعل عملية تطوير التطبيق سريعة للغاية بسبب ميزة إعادة التحميل السريع والتي تسمح لنا بمشاهدة تأثير أي تغيير في الشيفرة مباشرة.
- يشبه فلاتر الإطار التفاعلي حيث لا يحتاج المطورون الى تحديث محتوى واجهة المستخدم يدويا.
- أدوات ملائمة للمطورين، حيث وضعت Google سهولة الاستخدام في اعتبارها عند إنشاء Flutter، ومن خلال أدوات مثل "Hot Reload" (إعادة التحميل السريع)، بإمكان المطور معاينة المظهر الذي ستبدو عليه تغييرات التعليمة البرمجية بدون فقدان الحالة، وهو ما يجعل عملية تطوير التطبيق سريعة للغاية حيث يمكن مشاهدة تأثير أي تعديل في الشيفرة مباشرة بينما تسهّل أدوات أخرى، مثل Widget Inspector (فاحص الأدوات)، العرض المرئي للمشكلات المتعلقة بتخطيطات واجهة المستخدم وحلها.
- يقلل من وقت جهود الاختبار حيث تعمل تطبيقات فلاتر عبر الأنظمة الأساسية وبالتالي لا يحتاج المختبرين الى تشغيل نفس مجموعة الاختبارات على منصات مختلفة لنفس التطبيق.

- يحتوي واجهة مستخدم ممتازة لأنه يستخدم أدوات وعناصر جاهزة في التصميم وأدوات تطوير عالية وواجهات برمجة تطبيقات متقدمة بالإضافة للعديد من المميزات الأخرى
- مناسب لتطبيقات الحد الأدنى من المنتجات القابلة للتطبيق Minimum Viable Products (MVP) بسبب عملية التطوير السريعة والطبيعة المشتركة للأنظمة الأساسية.
- أداء تطبيقات قريب من الأصلي. يستخدم Flutter لغة البرمجة Dart، وينفذ التحويل البرمجي إلى تعليمة برمجية للآلة ومن ثم تفهم الأجهزة المضيفة هذه التعليمة البرمجية، ما يضمن أداء سريع وفعال.
- عرض سريع ومتسق وقابل للتخصيص، فبدلاً من الاعتماد على أدوات العرض الخاصة بأنظمة أساسية محددة، يستخدم Flutter مكتبة الجرافيك (Skia) المفتوحة المصدر من Google لعرض واجهة المستخدم، وهذا يمنح المستخدمين مرئيات متسقة بغض النظر عن المنصة التي يستخدمونها للوصول إلى التطبيق.

مكونات إطار فلاتر الرئيسية هي:

- لغة برمجة Dart
- محرك فلاتر
- المكتبات الأساسية
- عناصر (widgets) مخصصة

كل شيء في فلاتر هو عبارة عن widgets حيث يزود إطار فلاتر المبرمجين بمجموعة كبيرة من هذه العناصر التي تؤدي وظائف متنوعة داخل التطبيق.

تم الإعلان عن الإصدار الأول من فلاتر في عام 2015 تحت الاسم الرمزي (SKY) ولكن أول اصدار مستقر منه وهو (Flutter 1.0) تم إصداره في كانون الأول 2018

آخر اصدار هو (3.35.0) تم إصداره في أغسطس من العام الحالي 2025

وبالنسبة لنا سنستخدم الإصدار (3.29.0) في عملية التطوير

لغة البرمجة دارت (Dart):

دارت هي لغة برمجة مفتوحة المصدر للأغراض العامة تم تطويرها من قبل شركة جوجل (Google) وتدعم تطوير التطبيقات في كل من الخادم والعميل، ولكنها تستخدم على نطاق واسع لتطوير تطبيقات الاندرويد والآيفون وتطبيقات الويب، وانترنت الأشياء باستخدام إطار العمل فلاتر (Flutter).

تعتبر لغة دارت لغة غرضية التوجه ديناميكية تدعم مفاهيم البرمجة من الواجهات (Interfaces) والصفوف (Classes) والوراثة وتستخدم مفهوم المجموعات لتكرار هياكل البيانات.

أحد أهداف اللغة بأن تعمل على جميع متصفحات الويب المتقدمة والأجهزة المحمولة وصولاً إلى خوادم الويب وما يميز اللغة هو إمكانية كتابة برنامج ونشره على أجهزة أندرويد وآيفون دون الحاجة الى إعادة كتابة التطبيق بلغة أخرى.

تم اصدار لغة دارت في عام 2011 ولكنها أصبحت شائعة بعد عام 2015 وتحديدا في أغسطس 2018 مع الإصدار (2.0) ويعود ذلك بشكل رئيسي الى ازدياد شعبية إطار العمل فلاتر.

والاصدار الاحدث من مجموعة تطوير البرامج (Dart SDK) حتى تاريخه هو (3.9)

حديثا تم دمج مجموعة تطوير البرامج (Dart SDK) وتضمينها ضمن فلاتر (Flutter SDK) ولم يعد هناك حاجة لتثبيتها بشكل منفصل.

2.5.1- إطار العمل (Django) ولغة البرمجة (Python) لتطوير الواجهة الخلفية:

يعد جانغو إطار عمل ويب عالي المستوى، حر، مجاني، مفتوح المصدر، وله مجتمع مزدهر ونشط بالإضافة الى توثيق ممتاز.

هدف جانغو الأساسي تسهيل إنشاء مواقع الوب المعقدة المعتمدة على قواعد البيانات. تركز المنصة على قابلية إعادة الاستخدام وقابلية التوصيل (Pluggability) للعناصر، شيفرة أقل، خفض الاقتران، تطوير سريع، ومبدأ لا تكرر نفسك.

يتم استخدام لغة البرمجة بايثون في كل المنصة حتى بالنسبة للإعدادات، الملفات وأنماط البيانات.

وتوفر كذلك خيار استعمال واجهة إدارة قواعد البيانات (إنشاء – قراءة – تحديث – حذف)

يتبع جانغو أسلوب تطوير نموذج–قالب–عرض (MVT: Model–Template–Views)

أصدر جانغو للعموم في يوليو 2005 تحت رخصة بي إس دي، وفي يونيو 2008 أعلن عن إنشاء مؤسسة برنامج جانغو التي ستتولى تطوير جانغو في المستقبل.

واستمر إطار العمل جانغو بالنمو والتطور والتحسن مع كل اصدار جديد بإضافة وظائف واصلاحات وتضمين دعم للأنواع الجديدة من قواعد البيانات ومحركات القوالب والتخزين المؤقت واضافة دوال وأصناف العرض المعممة التي أدت الى تقليل مقدار الشيفرة البرمجية المطلوبة لتحقيق المهام البرمجية.

أحدث اصدار من إطار العمل جانغو هو الإصدار (5.2.5) تم إصداره بتاريخ 6 آب 2025 وهو الإصدار الذي سنستخدمه في تطوير خدمات الخلفية للنظام.

لماذا جانغو؟

لابد بداية من التطرق الى التوثيق الممتاز والمنظم والذي تم الاستفادة منه في هذا المشروع لتجاوز أي مشكلات او لإيجاد الحلول المناسبة حيث يمكن العودة اليه والبحث عن أي موضوع بكل سهولة مع دعمه للتوثيق متعدد الاصدارات.

يقدم جانغو مجموعة كبيرة من الميزات ومنها لوحة التحكم والإدارة المدمجة في جانغو والتي تناسب حالتنا كوننا سنقوم بتطوير تطبيق موبايل كواجهة أمامية ووجود هذه الميزة بشكل مدمج سمح لنا بالحصول على كل عمليات الادارة المطلوبة باستخدام الضبط المناسب لملف (admin.py) ومن ثم يمكن استخدامها بدمجها مع تطبيق الواجهة الامامية لنوفر إمكانية الاستخدام لمدير النظام من داخل التطبيق.

وبالتالي فان وجود لوحة تحكم بشكل افتراضي وكونها قابلة للتعديل والتخصيص بشكل كبير وفق حاجتنا سيوفر علينا الكثير من الجهد والوقت لبناء مثيلتها من الصفر.

وبالإضافة لما سبق تبرز أهمية جانغو من كونه يسمح ببناء برمجيات تتمتع بالمميزات التالية:

- **مكتملة:** يتبع جانغو فلسفة كل شيء مضمن (Batteries Included) حيث يوفر للمطورين كل ما يحتاجونه وكل ما يرغبون بتطويره.
- **متعددة الاستعمالات:** يمكن استخدام جانغو – وقد استخدم فعليا – في بناء أي نوع من مواقع الويب تقريبا بدءا من مواقع إدارة المحتوى والويكي الى شبكات التواصل الاجتماعي والمواقع الإخبارية، ويمكن لجانغو العمل مع أي إطار عمل من طرف العميل وتقديم المحتوى بأي تنسيق تقريبا.
- **آمنة:** يساعد جانغو على تجنب الأخطاء الأمنية الشائعة وحماية الموقع آليا فمثلا يؤمن تعمية لكلمات المرور (Hash) بدلا من تخزينها مباشرة وذلك بتمريرها في دالة تعمية مشفرة

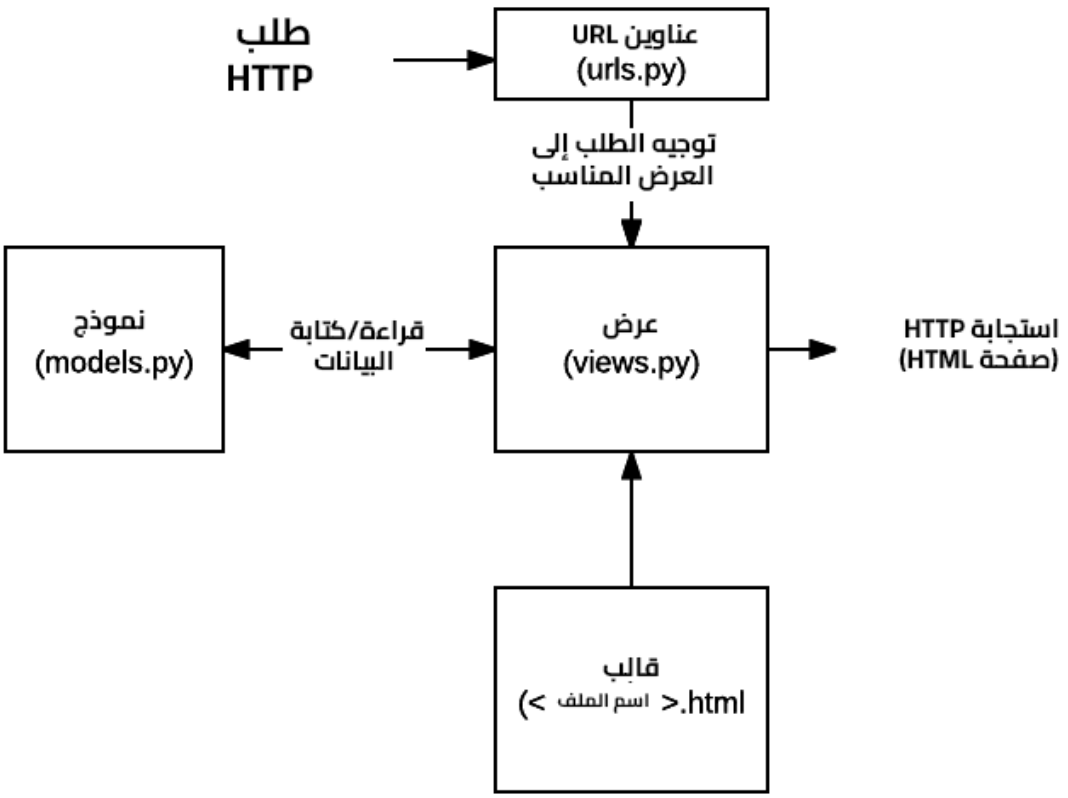
(Cryptographic Hash Function) ويقوم لاحقا عند تسجيل الدخول بتمرير كلمة المرور المدخلة عبر نفس الدالة ومقارنة الخرج مع قيمة التعمية المخزنة للتحقق من صحة كلمة المرور وبالتالي حتى لو تم الوصول الى قيمة التعمية فمن الصعب جدا للمهاجم معرفة كلمة المرور الاصلية نظرا لطبيعة الدالة أحادية الاتجاه، كما يؤمن جانغو الحماية ضد العديد من الثغرات افتراضيا مثل هجمات حقن استعلامات SQL وهجمات السكريبتات العابرة للمواقع وتزوير الطلبات عبر المواقع والاختطاف بالنقر وغيرها.

- **قابلة للتوسع:** يستخدم جانغو معمارية لا شيء مشترك القائمة على المكونات بحيث يكون كل جزء من المعمارية مستقل عن الأجزاء الأخرى بما يسمح باستبداله أو تغييره عند الحاجة، وبالتالي يمكن إضافة وإزالة التطبيقات الى المشروع بكل سهولة.
- **قابلة للصيانة:** تمت كتابة شيفرة جانغو باستخدام مبادئ وأنماط التصميم التي تشجع على انشاء شيفرة برمجية قابلة للصيانة وإعادة الاستخدام، ويتبع مبدأ (DRY) الذي يعني لا تكرر نفسك Don't Repeat Yourself مما يقلل بشكل كبير من كمية الشيفرة البرمجية.

يدعم جانغو أيضا تجميع الوظائف المرتبطة ببعضها ضمن تطبيقات قابلة لإعادة الاستخدام، وتجميع الشيفرة البرمجية المرتبطة ببعضها ضمن وحدات على مستوى أدنى باستخدام الأسلوب نموذج–قالب–عرض (MVT: Model–Template–Views).

- **قابلة للنقل:** باعتبار إطار العمل جانغو تمت كتابته باستخدام لغة البرمجة بايثون والتي تعمل على معظم المنصات، بالتالي يمكن تشغيل تطبيقاته على العديد من إصدارات ويندوز ولينوكس وماك، كما يدعم العديد من مزودي استضافة الويب جانغو بصورة جيدة ويؤمنون بنية أساسية محددة وتوثيق جيد لاستضافة مواقع جانغو.

ويمكن تلخيص طريقة عمل جانغو كما هو موضح في الشكل التالي (2–6)



الشكل1 طريقة عمل جانغو

لغة البرمجة بايثون (Python):

تم تطوير لغة البرمجة بايثون في معهد الرياضيات والمعلوماتية الهولندي (CWI) في مدينة أمستردام على يد جايدو فان روسم في أواخر ثمانينات القرن العشرين، وكان أول إعلان عنها في عام 1991م.

بايثون لغة برمجة عالية المستوى، مفتوحة المصدر، قابلة للتوسيع، متعددة الاستعمالات، سهلة التعلم حيث تعتبر من بين أسرع لغات البرمجة تعلماً وتتميز بمجتمعها النشط ولها مجموعة كبيرة من المكتبات البرمجية ذات الأغراض الخاصة وتم بناء العديد من أطر العمل باستخدامها ومنهم إطار العمل جانغو (Django) الذي سنستخدمه في بناء خدمات الخلفية وواجهات التخاطب البرمجية (API)

تعتبر بايثون لغة برمجة متعددة الأنماط الفكرية حيث تدعم كلا من البرمجة كائنية التوجه (OOP) والبرمجة المهيكلة دعمًا كاملاً، كما تدعم بايثون البرمجة الوظيفية والبرمجة جانبية المنحى بما في ذلك عن طريق البرمجة الوصفية والكائنات الوصفية «خاصةً الطرق»، وتدعم الوراثة العادية والمتعددة.

أحدث اصدار مستقر من اللغة حتى تاريخ 19 آب من عام 2025 هو (Python 3.13.7) وسنستخدم نحن في مشروعنا الاصدارة (3.10.11) ومن خلالها وباستخدام (PIP) مدير الحزم في بايثون سنقوم بتثبيت إطار العمل جانغو والمكتبات الملحقة الضرورية التي سنستخدمها في تطوير أجزاء هامة من النظام.

3.5.1- قاعدة البيانات (SQLite):

في البداية وأثناء مرحلة التطوير والاختبار سنقوم بالاعتماد على قاعدة البيانات المدمجة في جانغو وهي قاعدة بيانات (SQLite)

في المراحل اللاحقة يمكن الانتقال الى قواعد بيانات أكثر ملاءمة لبيئة الإنتاج مثل (MySQL, PostgreSQL)

لماذا SQLite؟

تعتبر محرك قاعدة بيانات صغير الحجم لا يتجاوز حجم مكتبتها 600 كيلوبايت وتأتي مدمجة في إطار العمل فلاتر فلا حاجة لتثبيت أي إضافات ومن ميزاتها أيضا انها (Serverless) أي أنها لا تتطلب عملية خادم منفصلة للتشغيل وتتم الكتابة والقراءة على الملف بشكل مباشر.

ميزة تخزين قاعدة البيانات في ملف واحد يجعلها ذات قابلية نقل عالية ويسهل عمليات النقل والنسخ الاحتياطي بين مختلف الأنظمة وبالتالي تعمل على نطاق واسع من أنظمة التشغيل.

يتم دوما تحسينها وتطويرها وتحقق أداء عال في عمليات القراءة وخصوصا في التطبيقات التي تتطلب استرجاع البيانات بسرعة.

تقدم دعم واسع لمعظم ميزات (SQL) القياسية وتستهلك ذاكرة ومساحة تخزين قليلة.

4.5.1- برمجيات التطوير:

قمنا باستخدام البرمجيات التالية في عملية التطوير:

- برنامج Visual Studio Code لكتابة الكود البرمجي.
- برنامج Android Studio لإجراء المحاكاة أثناء عملية التطوير.

5.5.1- خدمة الاشعارات:

تم تنفيذ خدمة الاشعارات بالاعتماد على الخدمة السحابية المقدمة من شركة جوجل (Firebase Cloud Messaging - FCM)

الفصل الثاني: تحليل النظام

1.2- مقدمة:

بالرغم من وجود عدة نماذج لعملية التطوير تم الاعتماد في عملية التطوير لتطبيق المساعد الجامعي الشخصي على نموذج النموذج الأولي باعتباره نموذج مبسط وقابل للاختبار لفكرة أو لمنتج، يستخدم لجمع الملاحظات والتكرار عليها قبل التطوير النهائي

ميزة هذا النموذج الاساسية أنه بدلا من تحديد (وبالتالي تجميد) المتطلبات قبل المضي الى التصميم والتنجيز يتم انشاء نموذج أولي لفهم المتطلبات يمكن العميل من الحصول على إحساس فعلي بالنظام وبالتالي توفير تغذية راجعة فورية تمكن من فهم المتطلبات بشكل أفضل.

مزايا النموذج الأولي

- **المشاركة:** يسمح هذا النظام للمستخدمين بالمشاركة بشكل فعال في عملية التطوير.
- **اكتشاف الأخطاء مبكراً:** حيث يسهم في تحديد المشكلات الوظيفية أو التصميمية خلال المراحل الأولى من التطوير، مما يقلل تكاليف الإصلاح لاحقًا.
- **اكتشاف الوظائف المفقودة:** ان مشاركة المستخدمين في عملية التطوير تمكن من التعرف على الوظائف المفقودة بسهولة.
- **تحسين تجربة المستخدم:** يسمح باختبار التصميم مع المستخدمين الفعليين وجمع ملاحظاتهم، مما يعزز وضوح الواجهات وسهولة الاستخدام.
- **تعزيز التواصل بين الفرق:** يوحد رؤية المصممين والمطورين وأصحاب المصلحة حول المنتج النهائي، ويقلل سوء الفهم.
- **تسريع وقت التسويق:** يقلل الحاجة إلى إعادة العمل عبر التكرار السريع، خاصةً في المشاريع المعقدة مثل التطبيقات التعليمية.
- **تقليل المخاطر المالية:** يجنب الاستثمار في ميزات غير ضرورية عبر التحقق من متطلبات المستخدم قبل التطوير الكامل.

التحديات المحتملة

- **تحويل النموذج إلى منتج نهائي:** قد يصبح النموذج الأولي أساسًا للمنتج النهائي دون تصميم معماري متين، مما يهدد الأداء.

- **استهلاك الموارد:** إذا لم يدار بعناية، قد يطيل الجدول الزمني بسبب التكرارات المتعددة.
- **توقعات غير واقعية:** قد يوهم المستخدمين بوجود منتج شبه كامل، بينما النموذج يقتصر على واجهات أو وظائف محدودة.
- **التوسع:** قد تزيد هذه المنهجية من تعقيد النظام حيث قد يتوسع نطاق النظام الى ما يتجاوز الخطط الاساسية

أفضل ممارسات للتطبيق

1. **البدء بنماذج منخفضة الدقة:** استخدام رسومات تخطيطية (Wireframes) أو نماذج ورقية لتجربة الأفكار الأساسية بسرعة.
2. **الاختبار مع المستخدمين المستهدفين:** تضمين طلاب من الجامعة في مرحلة الاختبار لجمع ملاحظات واقعية حول خدمات مثل الجداول الدراسية والمحتوى التعليمي.
3. **التكرار بناءً على الملاحظات:** تعديل النموذج أولاً بأول وفقًا لملاحظات المستخدمين، مع التركيز على تحسين التدفقات الرئيسية كالإشعارات وإدارة المحتوى.
4. **الانتقال التدريجي للدقة العالية:** تطوير نماذج تفاعلية (قابلة للنقر) بمجرد استقرار المتطلبات الأساسية.
5. **تحديد نطاق واضح:** تجنب إضافة ميزات غير ضرورية، والتركيز على الوظائف الأساسية كتلك المطلوبة في تطبيق "وسيط."

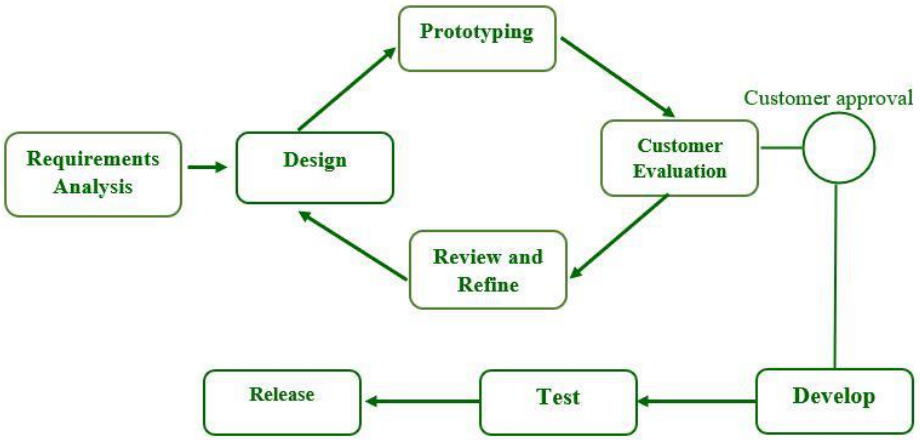
وبالتالي يعد النموذج الأولي الخيار الأمثل وفوائده في تقليل المخاطر وتحسين الجودة تفوق تحدياته عند تطبيقه باتباع أفضل الممارسات، مثل البدء بنماذج بسيطة والتكرار بناءً على ملاحظات المستخدمين.

مراحل نموذج البرمجة الأولي

للنموذج الأولي نمطين هما النمط المؤقت والنمط التطوري وفي حالتنا قمنا بالاعتماد على النموذج التطوري حيث يتم بناء نموذج أولي أولي، ثم يتم تطويره وتحسينه تدريجيًا حتى يتحول إلى المنتج النهائي. كل تكرار يبنى على ما قبله، لكن مع معالجة الاخطاء وإضافة مزيد من الوظائف والتحسينات بناءً على ملاحظات المستخدمين.

في نموذج البرمجة الأولي يقسم مسار التطوير إلى سلسلة من المراحل المتكررة تضمن حصول المنتج النهائي على التغذية الراجعة المبكرة وتحسين مستمر قبل إنتاج النسخة النهائية:

- مرحلة التحليل وجمع المتطلبات: وفيها يتم توثيق احتياجات المستخدمين وتحديد نطاق الوظائف المراد محاكاتها في النموذج الأولي.
- مرحلة التصميم السريع (Wireframing): وفيها يتم إنشاء مخططات هيكلية منخفضة الدقة لواجهة المستخدم تُوضح تدفقات الاستخدام الأساسية دون التركيز على الجوانب الجمالية.
- مرحلة بناء النموذج الأولي: يتم فيها تطوير نسخة أولية تفاعلية (قابلة للنقر أو تحتوي على وظائف جزئية) تحاكي بعض خصائص المنتج المتوقع.
- مرحلة الاختبار وجمع الملاحظات: عرض النموذج الأولي أمام عينة من المستخدمين النهائيين أو أصحاب المصلحة للحصول على تقييمهم وآرائهم حول سهولة الاستخدام وصحة تدفقات العمل.
- مرحلة التكرار والتحسين: تتم مراجعة التصميم والوظائف بناء على الملاحظات، ثم تحديث النموذج الأولي لتعزيز النقاط القوية ومعالجة النقائص.
- مرحلة التطوير النهائي: عندما يستقر النموذج الأولي ويحقق متطلبات الجودة، يبدأ فريق التطوير في بناء المنتج الحقيقي استنادًا إلى التصميم والمواصفات المحسّنة.
- مرحلة الصيانة والتطوير المستمر: بعد إطلاق المنتج، يستمر جمع التغذية الراجعة والتحديثات لضمان توافقه مع احتياجات المستخدم وسوق العمل على المدى الطويل.



الشكل 2 مخطط نموذج النموذج الأولي

يجب التنويه أيضا انه في بداية أي عملية تطوير لا بد من القيام بدراسة الجدوى ووضع خطة للمشروع.

الغرض من دراسة الجدوى هو تحديد إذا ما كان تنفيذ النظام المقترح سيدعم ويحقق اهداف المنظمة ويضمن انه قادر على تلبية احتياجات المستخدم وبحيث يكون فعال من حيث التكلفة.

وباعتبار ان المشروع قيد الإنجاز هو لغرض دراسي سنتجاوز مرحلة دراسة الجدوى ونبدأ فورا بالخطوة التالية في دورة حياة تطوير البرامج وهي مرحلة التحليل، موضوع هذا الفصل.

مرحلة التحليل:

يعتبر تحليل النظام الخطوة الأهم في دورة حياة تطوير البرامج حيث يتم فيها جمع البيانات الواقعية وفهم العمليات المطلوب برمجتها وتحديد المشكلة بدقة ثم تقديم مستند يشكل أساسا لتطوير البرنامج

عند بدء أي مشروع برمجي جديد هناك ثلاثة أطراف رئيسية تكون ضمن دائرة الاهتمام وهم: العميل، المطور، والمستخدمين.

وهنا يمكننا في هذا المشروع ان نمثل دور العميل والمطور لكن يبقى المستخدمين هم أساس النظام البرمجي وتلبية احتياجاتهم ورغباتهم هو الذي سيحدد نجاح المشروع

لذلك لابد من إيجاد طريقة لمعرفة هذه الاحتياجات والرغبات وتحديدها بدقة، وتجاوز المشكلة الأساسية في كون المستخدم عادة لا يفهم طبيعة البرامج ولا طبيعة عملية تطويرها، بالإضافة الى ان المطور أيضا من الممكن الا يفهم مشكلة المستخدمين او المنطق الذي سيكون على النظام تنفيذه.

تأتي عملية توصيف المتطلبات كحل مثالي يمكن الجميع من تكوين رؤية مشتركة حول البرنامج او النظام قيد البناء.

تتكون هذه المرحلة من ثلاثة أنشطة رئيسية:

- استيضاح المتطلبات
- مواصفات المتطلبات
- التحقق من المتطلبات

2.2- استيضاح المتطلبات (Requirements Elicitation):

المفهوم:

هي عملية جمع وفهم احتياجات ورغبات أصحاب المصلحة والمستخدمين للنظام المستهدف. الهدف هو اكتشاف ما يحتاجه النظام فعليًا، وليس فقط ما يظنه المطورون أو العملاء

في مرحلة استيضاح المتطلبات يتم التركيز على تحليل المشكلة والمتطلبات وفهمها.

ويبدأ تحليل المشكلة عادة بتحديد ما يسمى ببيان المشكلة (Problem Statement) على مستوى عال من التجريد، والغرض الأساسي هنا هو الوصول الى فهم شامل لماهية البرنامج او النظام.

ومن أفضل الطرق التي يمكن الاعتماد عليها لتحقيق ذلك:

- إجراء مقابلات مع المستخدمين.
- تنظيم ورش عمل وجلسات عصف ذهني.
- استخدام الاستبيانات والملاحظات الميدانية.

الأهمية:

تساعد هذه المرحلة في بناء فهم عميق للمشكلة أو الفرصة التي يعالجها النظام، وتحدد نطاق العمل بدقة، وتقلل من مخاطر سوء الفهم أو تجاهل متطلبات مهمة.

فهم المشكلة:

حتى نبدأ باستيضاح المتطلبات يلزمنا أولا فهم المشكلة وبالتالي قمنا بجمع بعض البيانات والمعلومات واجراء مقابلات واجتماعات مع مستخدمي النظام الحالي لتحليل الية العمل الحالية والوقوف على المشاكل والمعيقات والمقترحات وتوصلنا الى فهم جيد للنظام الحالي وقمنا بتحديد أهم مشاكله وهي:

- عدم وجود منصة واحدة تقدم الخدمات مجتمعة حيث تتوزع حاليا على ثلاث منصات هي (نظام معلومات وامتحانات الجامعة – نظام إدارة التعلم – نظام البريد الالكتروني)
- عدم وجود نظام للتنبيهات والاشعارات بخصوص الأحداث الهامة
- صعوبة تتبع المواعيد الهامة

- الحاجة دوما للدخول الى المواقع المختلفة لهذه الخدمات عبر المتصفح

الحل المقترح:

بعد فهم المشكلة والتعرف على نقاط الضعف والمعوقات الواجب معالجتها بدأنا بالتفكير في إيجاد الحل المناسب لها بحيث يكون متلائم مع الظروف الحالية وألا يقتصر فقط على حل هذه المشكلة وتفادي السلبيات بل يتجاوز ذلك الى ادخال تحسينات واضافات بالاستفادة من أحدث التقنيات المتوفرة بالشكل الأمثل.

سيعتمد الحل المقترح على بناء تطبيق للجوال يجمع اهم الخدمات التي تقدمها هذه المنصات والتي يحتاجها الطالب في منصة واحدة بحيث يمكنه عند فتح التطبيق الوصول الى ما يلي:

- اهم المعلومات الشخصية من نظام المعلومات المحدثة بشكل دائم.

- الموارد التعليمية وهنا سيتم تقديم الموارد ضمن فئتين الأولى هي الموارد الرسمية الموجودة على نظام إدارة التعلم والثانية هي الموارد الاضافية التي سيقوم المرشدين المتطوعين (Mentors) بإضافتها لتعزيز تجربة التعلم ومساعدة الطلاب على التوسع في الدراسة.

- روبوت دردشة مدمج يمكن استخدامه من داخل التطبيق

- عرض البريد الالكتروني مع تأمين نظام اشعارات بالوقت الحقيقي عند وصول بريد جديد.

- مجموعة من الأدوات المساعدة مثل نظام مذكرات وحاسبة ومنبه.

3.2- مواصفات المتطلبات (Requirements Specification):

المفهوم:

هي عملية توثيق المتطلبات التي تم جمعها في المرحلة السابقة بشكل رسمي ومنظم، بحيث تكون واضحة ومفهومة لجميع الأطراف.

ان الفهم الذي حصلنا عليه في عملية استيضاح المتطلبات من خلال تحليل المشكلة وتكوين صورة عامة عن النظام الحالي ومشاكله وما هو المأمول من النظام البديل سيكون الأساس في تحديد المتطلبات ومواصفاتها.

الخطوات والأساليب:

- اعداد مستند مواصفات المتطلبات (SRS)
- تصنيف المتطلبات إلى وظيفية (ما يجب أن يفعله النظام) وغير وظيفية (الجودة، الأداء، الأمان).
- استخدام الرسوم من مخططات ونماذج بالاستعانة بلغة النمذجة الموحدة (UML) والتي تسهل التخاطب بين التحليل والتصميم وتوضح العلاقات وحالات الاستخدام.
- تحديد المدخلات والمخرجات لكل وظيفة، والشروط المسبقة واللاحقة، والآثار الجانبية.

الأهمية:

توفر هذه الوثيقة مرجعية رسمية لفريق التطوير، وتساعد في توحيد الرؤية بين جميع الأطراف، وتقلل من التعديلات غير المخطط لها أثناء التنفيذ.

تقسم المتطلبات الى متطلبات وظيفية تحدد الوظائف التي يقدمها النظام ومتطلبات غير وظيفية تركز على كيفية عمل النظام مثل متطلبات الأداء والأمان والجودة

1.3.2- المتطلبات الوظيفية:

المتطلبات الوظيفية هي الوظائف والخدمات التي يقدمها النظام للمستخدمين وتحدد سلوك النظام عند التفاعل معه.

وبالتالي فإن الهدف من هذه المتطلبات هو التعريف بالنواحي الوظيفية في النظام، أي أنها تصف مدخلات ومخرجات النظام وكيف نقوم بتحويل المدخلات إلى المخرجات التي نرغب بها.

ولذلك لا بد في البداية ان نقوم بتحديد الممثلين او الفاعلين في النظام وماهي المتطلبات الوظيفية لكل منهم وتحديد حالات الاستخدام ومواصفاتها بشكل واضح ورسم مخططات حالات الاستخدام الموافقة.

- **الممثلين في النظام (Actors):**

لدينا في النظام الأدوار أو الممثلين الفاعلين الرئيسيين (Primary Actors):

- مدير النظام (Admin): وهو في حالتنا مطور النظام ويكون مسؤول عن برمجة النظام وصيانته وتطويره
- الطالب (Student): وهو مستخدم البرنامج ويكون أحد طلاب الجامعة
- المرشد (Mentor): وهو أحد الطلاب الحاليين أو السابقين في الجامعة

بالإضافة الى الممثلين الرئيسيين لدينا فاعلين ثانويين (Secondary Actor) وهم:

- خدمة الاشعارات من جوجل (Firebase Cloud Messaging – FCM)
- أنظمة الجامعة الثلاثة (SVU Systems) وهي (SVUIS, LMS, EMAIL)

- **الوظائف والخدمات التي سيقدمها النظام:**

يجب ان يقوم النظام عند عمله بتقديم الوظائف والخدمات الرئيسة التالية:

- تسجيل الدخول
- إدارة معلومات الطالب
- إدارة التنبيهات
- إدارة التعلم
- إضافة مورد تعليمي
- حذف مورد تعليمي
- خدمة روبوت الدردشة
- إدارة التحميلات والملفات
- إضافة طلب ارشاد
- إدارة المرشدين

- البريد الالكتروني
- إدارة الملاحظات
- التواصل
- عرض التقويم
- الإدارة
- ارسال الأخبار
- ارسال الاشعارات
- إدارة التنبيهات
- تسجيل الخروج

بعد تحديد الممثلين الفاعلين ووظائف النظام سنقوم بتحديد مواصفات حالات الاستخدام عن طريق ربط كل وظيفة يقدمها النظام بالممثلين الفاعلين.

ينتج لدينا حالات الاستخدام الرئيسية التالية:

1.3.2.أ- حالات الاستخدام في النظام:

1. تسجيل الدخول Login:

Table 2 – Login

Use Case Specification	
Use Case ID	UC01-login
Use Case Name	Log In
Actors	*Primary*: Admin, Student, Mentor *Secondary*: SVU Systems, FCM Service
Description	تسجيل دخول المستخدم الى النظام مع دعم الدخول من عدة اجهزة
Preconditions	- المستخدم طالب حالي أو سابق في الجامعة ولديه حساب جامعي نشط حالياً.

Post conditions	- يتم انشاء حساب في قاعدة بيانات التطبيق في حال كان المستخدم جديد او تحديث بياناته في حال كان مستخدم حالي. - يتم إضافة معرف خاص بخدمة الاشعارات عند تسجيل الدخول من جهاز مختلف جديد.
Main Success Scenario	- عند تشغيل التطبيق للمرة الأولى يطلب الاذونات الضرورية وتظهر واجهة شروط الاستخدام مع خياري الموافقة والرفض - تظهر واجهة تسجيل الدخول وتحوي حقول اسم المستخدم وكلمة المرور. - يقوم المستخدم بإدخال بياناته - يتحقق النظام من البيانات عبر أنظمة الجامعة - ينتقل الى الواجهة الرئيسية ويعرض فيها معلومات الطالب
Exceptions	- عدم الموافقة على الشروط والاحكام (يخرج من التطبيق) - خطأ في البيانات المدخلة (عرض رسالة خطأ مع إمكانية إعادة المحاولة)
Assumptions	- الربط مستقر مع أنظمة الجامعة وتعمل بشكل طبيعي. - توافر اتصال انترنت. - التطبيق مثبت على جهاز المستخدم.

2. إدارة معلومات الطالب Student Management:

Table 3 - Student Management

Use Case Specification	
Use Case ID	UC02-student-management
Use Case Name	Student Management

Actors	*Primary*: Admin, Student, Mentor *Secondary*: SVU Systems
Description	إدارة معلومات الطالب وعرض كل التفاصيل في الواجهة الرئيسية للتطبيق
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح
Post conditions	- يتم عرض أحدث المعلومات في الصفحة الرئيسية
Main Success Scenario	- تفتح الواجهة الرئيسية ويعرض فيها معلومات الطالب المحدثة بشكل دائم في عدة اقسام مثل اسم البرنامج، المعدل التراكمي والوصفي والرقمي. - يتم عرض جدول مواعيد تظهر فيه الجلسات المجدولة والحالية مع مواعيدها. - يمكنه من هذه الصفحة عرض الجلسات، مواعيد الامتحانات ونتائج امتحانات المواد بقسميها العملي والنظري، عرض الصفوف، الدفعات المالية، جداول المحاضرات. - بالضغط على صورة المستخدم يتم عرض معلومات الملف الشخصي
Exceptions	
Assumptions	- الربط مستقر مع أنظمة الجامعة وتعمل بشكل طبيعي. - توافر اتصال انترنت.

3. إدارة التنبيهات Alarm Management:

Table 4 - Alarm Management

Use Case Specification

Use Case ID	UC03-alarm-management
Use Case Name	Alarm Management
Actors	*Primary*: Admin, Student, Mentor
Description	إدارة المنبهات في التطبيق سواء تنبيهات المحاضرات المجدولة او منبهات المستخدم الشخصية.
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح
Post conditions	- يتم إضافة او حذف التنبيه
Main Success Scenario	- من الواجهة الرئيسية يتم الضغط على الزر العائم رمز المنبه - تظهر واجهة إدارة التنبيهات وتحتوي عل التنبيهات النشطة. - في القسم العلوي نجد التنبيهات التلقائية للمحاضرات والتي ينشئها النظام تلقائيا عند وجود محاضرة للطالب ويتحكم الطالب في تفعيلها او الغاء تفعيلها. - في القسم التالي نجد تنبيهات المستخدم وهي التنبيهات الشخصية التي ينشئها المستخدم بشكل يدوي. - في أسفل يمين الواجهة يوجد زر عائم لإضافة تنبيه بالضغط عليه تفتح واجهة إضافة تحتوي على حقول اسم المنبه رسالة التنبيه وخانات اختيار التاريخ والوقت للمنبه وبإدخال القيم والضغط على "Add Alarm" يتم إضافة التنبيه الى مجموعة تنبيهات المستخدم. - في أسفل الواجهة يوجد خيار لإيقاف كل التنبيهات النشطة.

4. إدارة التعلم Learn Management:

Table 5 - Learn Management

Use Case Specification	
Use Case ID	UC04-learn-management
Use Case Name	Learn Management
Actors	*Primary*: Admin, Student, Mentor *Secondary*: SVU Systems
Description	إدارة التعلم باستخدام الموارد الجامعية والموارد الاضافية وروبوت الدردشة.
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح
Post conditions	- يتم عرض المحتوى المحدث
Main Success Scenario	- من الواجهة الرئيسية ننتقل الى الواجهة الثانية بالضغط على رمز إدارة التعلم في شريط التنقل السفلي لتظهر واجهة ادارة التعلم - في أعلى الصفحة القسم الخاص بنظام LMS الرسمي وفيه تظهر جميع المواد التي قام الطالب بالتسجيل عليها مع محتواها العلمي الخاص بالجامعة. - يظهر تحته القسم الخاص بالموارد الإضافية وفيه تظهر أسماء المقررات التي لدى الطالب وبالدخول الى مقرر محدد تظهر الموارد الإضافية الخاصة به في حال قام المرشد بإضافة موارد. - يمكن للطالب ان يقوم بتحميل الملفات من أي من القسمين وحفظها في جهازه للعودة اليها لاحقا حتى في حال عدم وجود اتصال بالإنترنت. - يظهر في الواجهة زر عائم لبدء دردشة مع روبوت الدردشة عند الضغط عليه تفتح واجهة الدردشة

Exceptions	- عدم وجود موارد إضافية مضافة (تظهر رسالة توضح ذلك). - خطأ في جلب البيانات (تظهر رسالة توضح ذلك ويقوم المستخدم بتحديث الصفحة)
Assumptions	- الربط مستقر مع أنظمة الجامعة وتعمل بشكل طبيعي. - الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي.

5. إضافة مورد تعليمي Add Resource:

Table 6 - Add Resource

Use Case Specification	
Use Case ID	UC05-add-resource
Use Case Name	Add Resource
Actors	*Primary*: Mentor Secondary: FCM Service
Description	إضافة موارد تعليمية متعلقة بمواضيع الكورسات التي يدرسها الطالب.
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح - المستخدم له دور مرشد (Mentor)
Post conditions	- يتم إعطاء الصلاحيات بالإضافة والحذف وظهور إضافات الواجهة المناسبة
Main Success Scenario	- من واجهة ادارة التعلم قسم الموارد الإضافية تظهر أسماء الكورسات الخاصة بالبرنامج وبالدخول الى أي منها نجد في اعلى الصفحة خيار إضافة مورد عند الضغط عليه تظهر الحقول الخاصة بتفاصيل المورد المراد اضافته من نوع واسم ووصف وملاحظات ورابط. - في حال كان المورد من النوع ملف تظهر خانة إضافية لإضافة الملف كمرفق. - يضغط على زر الإضافة فيتم إضافة المورد الجديد.

	- يتم ارسال اشعار الى جميع الطلاب المشتركين في هذا الكورس.
Exceptions	- خطأ في تعبئة البيانات (تظهر رسالة توضح ذلك)
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي.

6. حذف مورد تعليمي Remove Resource:

Table 7 - Remove Resource

Use Case Specification	
Use Case ID	UC06-remove-resource
Use Case Name	Remove Resource
Actors	*Primary*: Mentor
Description	حذف مورد تعليمي مضاف سابقاً
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح - المستخدم له دور مرشد (Mentor)
Post conditions	- يتم إعطاء الصلاحيات بالإضافة والحذف وظهور إضافات الواجهة المناسبة
Main Success Scenario	- من واجهة ادارة التعلم قسم الموارد الإضافية تظهر أسماء الكورسات الخاصة بالبرنامج وبالدخول الى أي منها نجد في اعلى الصفحة خيار إضافة مورد وفي حال وجود موارد مضافة سابقا تظهر تحته

	- في بطاقة كل مورد نجد زر حذف يسمح بحذف المورد عند الضغط عليه.
Exceptions	- عدم وجود موارد إضافية مضافة (تظهر رسالة توضح ذلك).
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي.

7. خدمة روبوت الدردشة Chat Bot:

Table 8 - Chat Bot

Use Case Specification	
Use Case ID	UC07-chatbot
Use Case Name	Chat Bot
Actors	*Primary*: Admin, Student, Mentor *Secondary*: Gemini Service
Description	استخدام روبوت الدردشة
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح
Post conditions	
Main Success Scenario	- من الواجهة الرئيسية ننتقل الى الواجهة الثانية بالضغط على رمز إدارة التعلم في شريط التنقل السفلي لتظهر واجهة ادارة التعلم - يظهر في الواجهة زر عائم لبدء دردشة مع روبوت الدردشة عند الضغط عليه تفتح واجهة الدردشة

	- يقوم المستخدم بكتابة أي سؤال في خانة السؤال ويضغط ارسال - يقوم البوت بتوليد الإجابة المناسبة واعادتها ضمن المحادثة
Exceptions	- خطأ في جلب البيانات (تظهر رسالة توضح وجود خطأ)
Assumptions	- الربط مستقر مع خدمات (Gemini) والخدمة تعمل بشكل طبيعي. - المستخدم متصل بالإنترنت باستخدام برنامج VPN بسبب حجب الخدمة عن سوريا حتى اللحظة

8. إدارة التحميلات والملفات Downloads Management:

Table 9 - Downloads Management

Use Case Specification	
Use Case ID	UC08-downloads-management
Use Case Name	Downloads Management
Actors	*Primary*: Admin, Student, Mentor
Description	إدارة التحميلات والملفات
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح. - المستخدم اعطى الاذونات المناسبة للتطبيق.
Post conditions	

Main Success Scenario	- في قسم موارد LMS الرسمي تظهر أسماء الكورسات الخاصة بالطالب وبالدخول الى أي منها تظهر الملفات التعليمية الخاصة بالجامعة وبالضغط على أي ملف يتم تحميله بشكل تلقائي وحفظه ضمن ذاكرة تخزين الجوال ضمن مجلد باسم المقرر. - في قسم الموارد الإضافية تظهر أسماء الكورسات الخاصة بالطالب وبالدخول الى أي منها تظهر الموارد الإضافية على شكل بطاقات وفي حال كان المورد من النوع ملف وسبق تحميله يظهر زر فتح الملف اما إذا لم يتم تحميله سابقا يظهر زر تحميل وبالضغط عليه يبدأ التحميل ويتم حفظه ضمن ذاكرة تخزين الجوال ضمن مجلد باسم المقرر مع سابقة كلمة "Resources" - من واجهة إدارة التعلم يتم الدخول الى الملفات المحملة بالضغط على زر التنزيلات الموجود اعلى يمين الواجهة فتظهر كل الملفات المحملة على شكل مجلدات مقسمة بحسب المواد
Exceptions	- عدم وجود تحميلات (تظهر رسالة توضح ذلك).
Assumptions	- الربط مستقر مع أنظمة الجامعة وتعمل بشكل طبيعي. - الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي.

9. إضافة طلب ارشاد Add Mentor Request:

Table 10 - Add Mentor Request

Use Case Specification	
Use Case ID	UC09-add-mentor-request
Use Case Name	Add Mentor Request
Actors	*Primary*: Student

Description	تقديم طلب الارشاد في برنامج معين
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح
Post conditions	- يتم إعطاء صلاحيات المرشد من إضافة وحذف وظهور عناصر وإضافات الواجهة المناسبة
Main Success Scenario	- من أي واجهة في التطبيق يتم فتح القائمة الجانبية (Drawer) - يختار الخيار "Request to be a mentor" فتفتح واجهة تحتوي حقول الادخال التالية: رسالة، رقم الهاتف - يقوم المستخدم بتعبئة البيانات المطلوبة والضغط على ارسال.
Exceptions	- خطأ في تعبئة البيانات (تظهر رسالة توضح ذلك)
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي.

10. إدارة المرشدين Mentors Management:

Table 11 - Mentors Management

Use Case Specification	
Use Case ID	UC10-mentors-management
Use Case Name	Mentors Management
Actors	*Primary*: Admin
Description	إدارة المرشدين في النظام وقبول او رفض الطلبات وتغيير حالة الارشاد للمستخدم
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح

Post conditions	- يتم تغيير حالة الارشاد اما تفعيل او الغاء وبالتالي تغيير صلاحيات المستخدم.
Main Success Scenario	- من أي واجهة في التطبيق يتم فتح القائمة الجانبية (Drawer) - تظهر قائمة المشرف او مدير النظام (Admin) وفيها الخيار "Mentor Requests" وبالضغط عليه تفتح واجهة إدارة الارشاد. - تظهر في هذه الواجهة قائمة الطلبات المقدمة من قبل الطلاب المهتمين بالانضمام الى طاقم المرشدين. - يمكن للمدير قبول او رفض أي طلب او تغيير حالة مرشد الى طالب فقط وبالعكس من خلال زر ضمن كل طلب - تظهر أيضا ضمن كل طلب الرسالة التي أرسلها مقدم الطلب ورقم هاتفه في حال ضرورة التواصل
Exceptions	
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي.

11. البريد الالكتروني Email:

Table 12 - Email

Use Case Specification	
Use Case ID	UC11-email
Use Case Name	Email
Actors	*Primary:* Admin, Student, Mentor *Secondary:* SVU Systems, FCM Service
Description	خدمة البريد الالكتروني استقبال وعرض الرسائل مع تخزينها محليا لإتاحتها في حال عدم وجود اتصال

Preconditions	- المستخدم قام بتسجيل الدخول بنجاح
Post conditions	- استقبال الرسائل وتخزينها محليا او إضافة الرسائل الجديدة للتخزين.
Main Success Scenario	- من الواجهة الرئيسية ننتقل الى الواجهة الثالثة بالضغط على رمز البريد الالكتروني الموجود في شريط التنقل السفلي لتظهر واجهة البريد الالكتروني - يتم عرض رسائل البريد الالكتروني الخاص بالطالب على شكل قائمة - يمكن الضغط على أي رسالة لعرض المحتوى الكامل مع التفاصيل. - يعرض التطبيق دوما اخر الرسائل حتى في وضع عدم الاتصال.
Exceptions	
Assumptions	- الربط مستقر مع أنظمة الجامعة وتعمل بشكل طبيعي.

12. إدارة الملاحظات Notes:

Table 13 - Notes

Use Case Specification	
Use Case ID	UC12-notes
Use Case Name	Notes
Actors	*Primary:* Admin, Student, Mentor
Description	إدارة الملاحظات
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح

Post conditions	- حفظ الملاحظة في قاعدة بيانات التطبيق
Main Success Scenario	- من الواجهة الرئيسية نضغط على الرمز الرابع في شريط التنقل السفلي وهو التطبيقات الإضافية (Apps) تفتح واجهة فيها مجموعة التطبيقات الإضافية نختار منها الملاحظات (Notes) - يفتح التطبيق الإضافي المفكرة وفيه يتم عرض الملاحظات المخزنة على شكل قائمة ويمكن بالضغط على زر "+" انشاء ملاحظة جديدة يكتب عنوان للملاحظة ومحتوى ثم يضغط حفظ - يمكن حذف ملاحظة محفوظة او تعديلها. - يمكن انشاء مجلدات لتنظيم الملاحظات ضمن تصنيفات يحددها المستخدم - يمكن الضغط على أي رسالة لعرض المحتوى الكامل مع التفاصيل او التعديل عليها.
Exceptions	
Assumptions	- المستخدم اعطى الاذونات المناسبة للتطبيق.

13. التواصل Contact Us:

Table 14 - Contact Us

Use Case Specification	
Use Case ID	UC13-contact-us
Use Case Name	Contact Us
Actors	*Primary:* Student, Mentor
Description	التواصل ومن خلاله يمكن للمستخدم التواصل مع المطورين للإبلاغ عن مشكلة او لطلب إضافة ميزة

Preconditions	- المستخدم قام بتسجيل الدخول بنجاح
Post conditions	- حفظ رسالة التواصل في قاعدة بيانات التطبيق - ارسال اشعار لمدير النظام
Main Success Scenario	- من أي واجهة في التطبيق يتم فتح القائمة الجانبية (Drawer) - يختار الخيار "Contact Us" فتفتح واجهة تحتوي حقول الادخال التالية: محتوى الرسالة، التصنيف - يقوم المستخدم بتعبئة البيانات المطلوبة والضغط على ارسال.
Exceptions	
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي.

14. عرض التقويم Calendar:

Table 15 - Calendar

Use Case Specification	
Use Case ID	UC14-calendar
Use Case Name	Calendar
Actors	*Primary:* Student, Mentor *Secondary*: SVU Systems
Description	عرض التقويم الخاص بالجامعة
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح
Post conditions	

Main Success Scenario	- من الواجهة الرئيسية نضغط على الرمز الرابع في شريط التنقل السفلي وهو التطبيقات الإضافية (Apps) تفتح واجهة فيها مجموعة التطبيقات الإضافية نختار منها التقويم (Callender) - يفتح التقويم الخاص بالجامعة ضمن التطبيق - -
Exceptions	
Assumptions	- الربط مستقر مع أنظمة الجامعة وتعمل بشكل طبيعي.

15. الإدارة Administration:

Table 16 - Administration

Use Case Specification	
Use Case ID	UC15-administration
Use Case Name	Administration
Actors	*Primary*: Admin *Secondary*: FCM Service
Description	عمليات الإدارة في النظام
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح. - المستخدم له الدور مدير النظام
Post conditions	

Main Success Scenario	- من أي واجهة في التطبيق يتم فتح القائمة الجانبية (Drawer) - تظهر مجموعة خيارات الادارة. - من الخيار (Dashboard) يمكن للمدير الدخول الى لوحة تحكم النظام والتعديل على قاعدة البيانات - من الخيار (Mentor Requests) يقوم المدير بمراجعة طلبات الارشاد والاستجابة بالرفض او الموافقة. - من الخيار (Send Notification) يمكن للمدير ارسال اشعار جماعي لكل مستخدمي التطبيق. - من الخيار (Send News) يمكن للمدير ارسال الاخبار لكل مستخدمي التطبيق.
Exceptions	
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي. - خدمة الاشعارات تعمل بشكل طبيعي.

16. إرسال الأخبار Send News:

Table 17 - Send News

Use Case Specification	
Use Case ID	UC16-send-news
Use Case Name	Send News
Actors	*Primary*: Admin *Secondary*: FCM Service
Description	إرسال الأخبار لمستخدمي التطبيق
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح. - المستخدم له الدور مدير النظام

Post conditions	- يتم إضافة الخبر الى قاعدة البيانات وارسال اشعار لجميع المستخدمين
Main Success Scenario	- من أي واجهة في التطبيق يتم فتح القائمة الجانبية (Drawer) - تظهر مجموعة خيارات الادارة. - يتم الضغط على الخيار (Send News) - يقوم بتحديد مستوى الأولوية (High, Medium, Low) - يقوم بتحديد تصنيف الخبر (خاص بالجامعة – خاص بالتعليم – عام ...) - يقوم بإدخال عنوان للخبر - يقوم بإدخال وصف للخبر - يقوم بوضع رابط الخبر الأصلي - يضغط ارسال - يتم إضافة الخبر الى بداية الاخبار عند جميع المستخدمين مع ارسال اشعار.
Exceptions	
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي. - خدمة الاشعارات تعمل بشكل طبيعي.

17. إرسال الإشعارات Send Notification:

Table 18 - Send Notification

Use Case Specification	
Use Case ID	UC17-send-notification
Use Case Name	Send Notification
Actors	*Primary*: Admin *Secondary*: SVU Systems, FCM Service

Description	ارسال اشعار جماعي لكل مستخدمي التطبيق.
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح. - المستخدم له الدور مدير النظام
Post conditions	- يتم ارسال اشعار لجميع المستخدمين النشطين عبر خدمة اشعارات فيربيس.
Main Success Scenario	- من أي واجهة في التطبيق يتم فتح القائمة الجانبية (Drawer) - تظهر مجموعة خيارات الادارة. - يتم الضغط على الخيار (Send Notification) - يقوم بإدخال عنوان للإشعار. - يقوم بإدخال محتوى الاشعار. - يضغط ارسال - يتم ارسال اشعار بالعنوان والمحتوى المدخل الى جميع المستخدمين النشطين.
Exceptions	
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي. - خدمة الاشعارات تعمل بشكل طبيعي.

18. تسجيل الخروج Log out:

Table 19 - Log Out

Use Case Specification	
Use Case ID	UC19-logout
Use Case Name	Log Out
Actors	*Primary*: Admin, Student, Mentor *Secondary*: FCM Service

Description	تسجيل خروج المستخدم من التطبيق
Preconditions	- المستخدم قام بتسجيل الدخول بنجاح.
Post conditions	- يتم إزالة بيانات الجلسة وحذف معرف خدمة الاشعارات FCM من قاعدة بيانات Firebase ومن الكائن المرتبط بهذا الطالب في قاعدة البيانات
Main Success Scenario	- من أي واجهة في التطبيق يتم فتح القائمة الجانبية (Drawer) - يختار الخيار "Log Out" فيظهر مؤشر الانتظار حتى تتم العملية ليعود التطبيق الى واجهة تسجيل الدخول
Exceptions	
Assumptions	- الربط مستقر مع خدمات الخلفية وتعمل بشكل طبيعي.

1.3.2.ب- مخطط حالات الاستخدام:

مخطط حالات الاستخدام هو عبارة عن تمثيل بصري يظهر فيه كيفية تفاعل المستخدمين (أو الأنظمة الخارجية) مع نظام معين لتحقيق أهداف محددة.

تؤمن لنا لغة النمذجة الموحدة (UML) إمكانية انشاء مخططات الاستخدام (Use Case Diagrams) والتي تعد أداة أساسية في مرحلة تحليل المتطلبات وتصميم الأنظمة في هندسة البرمجيات.

الغرض الرئيسي من مخطط حالات الاستخدام هو توفير نظرة عامة عالية المستوى لوظائف النظام من منظور المستخدم، وتحديد ما يفعله النظام دون الدخول في تفاصيل حول كيفية قيامه بذلك.

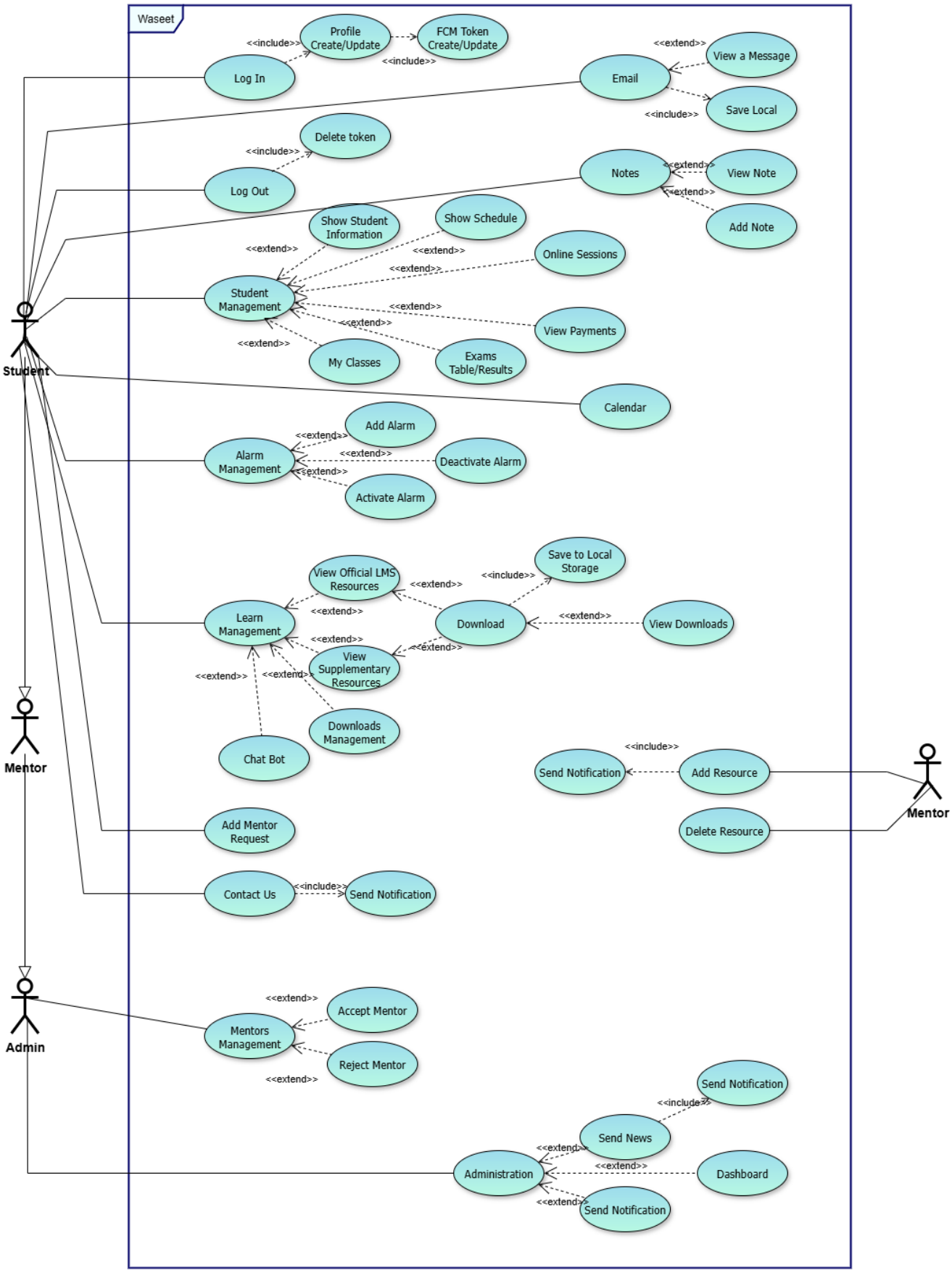
يساعد هذا المخطط على فهم المتطلبات الوظيفية للنظام، وتحسين التواصل بين المطورين وأصحاب المصلحة، وتوجيه عملية التطوير.

يتكون المخطط عادة من:

- الفاعلون (Actors): وهم الأشخاص او الأنظمة الخارجية التي تتفاعل مع النظام
- حالات الاستخدام (Use Cases): وتمثل على شكل بيضوي بداخله اسم الحالة وعادة تصف الاجراء او الوظيفة التي يمكن القيام بها او التي يقدمها النظام.
- حدود النظام (System Boundaries): وهو عبارة عن مربع او مستطيل يحيط بحالات الاستخدام ليحدد نطاق النظام وحدوده وبالتالي يفصل ما بين داخل النظام وخارجه
- العلاقات (Relationships): وهي التي توضح كيفية ارتباط الفاعلين بحالات الاستخدام او العلاقات ما بين حالات الاستخدام مثل التضمين (include) والتوسع (extend).

وباعتبار قد قمنا مسبقا بتحديد مواصفات حالات الاستخدام يمكننا الان باستخدام أي برنامج يدعم لغة النمذجة الموحدة رسم مخطط حالات الاستخدام لهذا النظام.

هناك مجموعة من البرامج التي يمكن استخدامها وهنا سنقوم باستخدام (drawio) الذي يقدم خدمة مجانية للتصميم على السحابة وبواجهة ممتازة تحوي جميع الأدوات اللازمة.



الشكل3 مخطط حالات الاستخدام في النظام

2.3.2- المتطلبات غير الوظيفية:

المتطلبات غير الوظيفية هي القيود والمعايير التي يلتزم بها النظام لضمان جودته وكفاءته، وتشمل جوانب الأداء والأمان وقابلية الاستخدام وغيرها.

- **متطلبات الأداء:**

- تحقيق وقت استجابة قليل للعمليات الرئيسية (عرض الجدول، فتح المحاضرات) يؤمن تجربة مستخدم سلسة
- دعم التخزين المحلي لتأمين إمكانية الاستخدام في حال عدم توافر اتصال بالإنترنت.
- العمل بكفاءة ودقة من خلال التعامل مع كل حالات الأخطاء المحتملة بحيث لا تؤثر على تجربة المستخدم وسير العمل
- القابلية للتحديث والتطوير وإمكانية إضافة خصائص وميزات إضافية لاحقا.
- ضمان العمل المستقر وتوفير الخدمة بشكل دائم.
- دعم إصدارات أندرويد المختلفة والتوافق مع أحدث الإصدارات.

- **الأمان والخصوصية**

- تحديد الصلاحيات والوصول بحسب كل دور.
- تخزين آمن للبيانات الحساسة سواء على الجهاز أو على السيرفر مع حماية من الوصول غير المصرح به.
- الامتثال لسياسة الخصوصية وحماية بيانات الطلاب.

- **الصيانة والتطوير المستمر**

  - اعتماد بنية برمجية معيارية (Modular Architecture) لتسهيل دمج تحديثات مستقبلية.
  - توثيق الكود والواجهات البرمجية (APIs) لضمان سهولة متابعة التطوير.

- **متطلبات الواجهة الخارجية:**

باعتبار ان التطبيق يعتمد بشكل كبير على تفاعل المستخدم لذلك لا بد من ان تكون الواجهات واضحة وبسيطة وسهلة الاستخدام ولتأمين التفاعلات المطلوبة وضمان تجربة مستخدم سلسة وفعالة يجب مراعاة النواحي التالية:

1. التصميم:

من المهم ان يتوافق التصميم مع أحدث إرشادات تصميم أندرويد (Material Design) لضمان تجربة موحدة واختيار ألوان وخطوط متناسقة وجذابة ومريحة مع ايقونات معبرة بتصميم بسيط

2. الاستجابة والتكيف:

يجب ان تكون الواجهة متجاوبة وتعمل بشكل جيد على مختلف الأجهزة بمختلف احجام الشاشات وعلى الأجهزة اللوحية وبحيث تدعم تدوير الشاشة لتعمل بشكل مناسب في الوضعين الأفقي والعمودي.

3. سهولة الاستخدام:

تأمين تجربة استخدام موحدة وسهلة التنقل بين الخدمات دون الحاجة لخبرة تقنية مسبقة بحيث تكون عناصر التنقل وازرار التفاعل واضحة وتتفاعل بشكل واضح مع اللمس

- **القيود**

تتمثل قيود هذا النظام بالنقاط الآتية:

- يعمل هذا النظام كمساعد تعليمي لطلاب الجامعة الافتراضية السورية.
- سيتم الاعتماد فقط على البرمجيات مفتوحة المصدر في تطوير النظام.
- ستقتصر الواجهة الأمامية حالياً على تطبيق الجوال.
- تطبيق جوال موحد لكل المستخدمين وتتم إدارة الأدوار في التطبيق.
- تطبيق الجوال يعمل على أجهزة الجوال الحاملة لنظام التشغيل أندرويد.
- يمكن ان تعمل بعض ميزات التطبيق بدون وجود انترنت ولكن يلزم وجود اتصال بالإنترنت لتعمل بعض الخدمات في التطبيق التي تعتمد على جلب البيانات في الزمن الفعلي.

4.2- التحقق من المتطلبات (Requirements Verification):

المفهوم:

هي عملية مراجعة وفحص المتطلبات الموثقة للتأكد من أنها كاملة، دقيقة، قابلة للتحقيق، ومتسقة

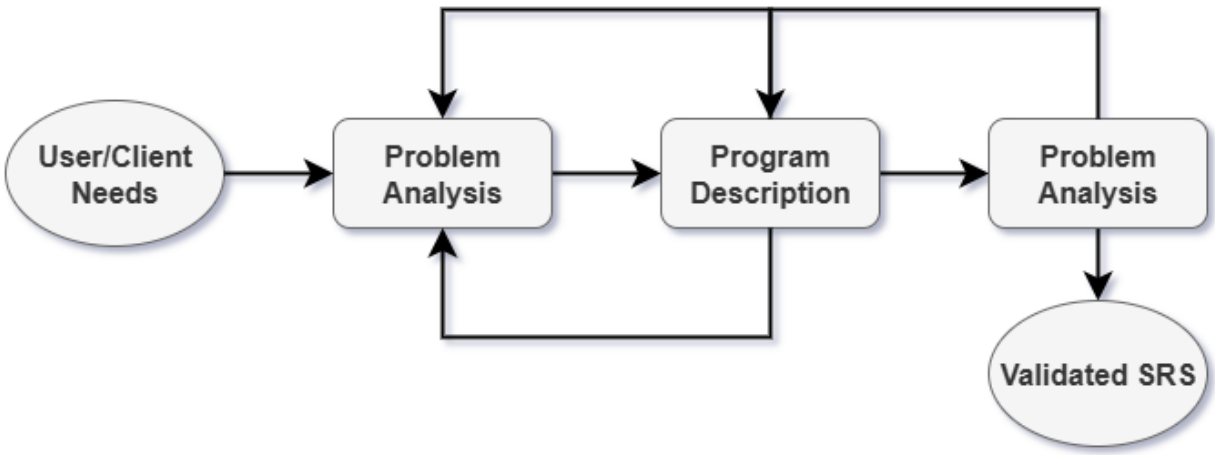
مع أهداف المشروع وبالتالي التأكد ان المتطلبات المذكورة هي فعلا المطلوبة وتم فهمها بالشكل الصحيح.

الخطوات والأساليب:

- الاجتماع مع أصحاب المصلحة والمستخدمين لمراجعة وثيقة المتطلبات.
- إجراء اختبارات تحقق (مثل مراجعات رسمية، حالات اختبار أولية).
- التأكد من عدم وجود تعارضات أو غموض في المتطلبات.
- التأكد من أن كل متطلب قابل للقياس والتنفيذ ضمن حدود المشروع.

الأهمية:

تضمن هذه المرحلة أن النظام الذي سيتم تطويره سيحقق فعلاً احتياجات المستخدمين، وتقلل من مخاطر إعادة العمل أو ظهور مشاكل أثناء مراحل التصميم والتطوير.



الشكل4 اجرائية التحقق من المتطلبات

الفصل الثالث: تصميم النظام

مقدمة:

بعد انتهاء مرحلة تحليل النظام وجمع المتطلبات وتحديد مواصفاتها لا بد من الانتقال الى المرحلة التالية والتي تتضمن تحويل هذه المتطلبات الى هيكل يكون مناسب للتنفيذ باستخدام احدى لغات

البرمجة حيث يتم وضع تصميم للنظام من كل جوانبه وتحديد معمارية مناسبة قبل الانتقال الى مرحلة التنفيذ التي تتضمن تحول التصميم الى أسطر برمجية.

في مرحلة التصميم تم الاعتماد على نهج التصميم الموجه للكائنات (Object Oriented Design – OOD) والذي يتعامل مع النظام كمجموعة من الكيانات او الكائنات التي تدير كل منها معلومات الحالة الخاصة بها.

وباعتبار اننا سنستخدم بايثون ودارت في عملية التنفيذ سنتمكن من الالتزام بهذا النهج كون لغتي البرمجة آنفتي الذكر تدعم البرمجة الكائنية.

لنبدأ أولا بالحديث عن تصميم قاعدة البيانات المناسب لتطوير تطبيقنا بشيء من التفصيل

1.3- تصميم قاعدة البيانات (Database Design):

1.1.3- مقدمة:

تعد قواعد البيانات مكون أساسي ومهم لأي مشروع برمجي فهي بمثابة مخزن منظم للبيانات يؤمن عمليات التخزين والإدارة والاسترجاع بكفاءة لذلك يمكن تشبيهها بالعمود الفقري للتطبيق والتصميم غير الجيد لقاعدة البيانات سينعكس حكما على أداء وفعالية البرنامج وقد يزيد من تعقيد عملية البرمجة حتى بالإضافة الى مشاكل الأمان وموثوقية البيانات

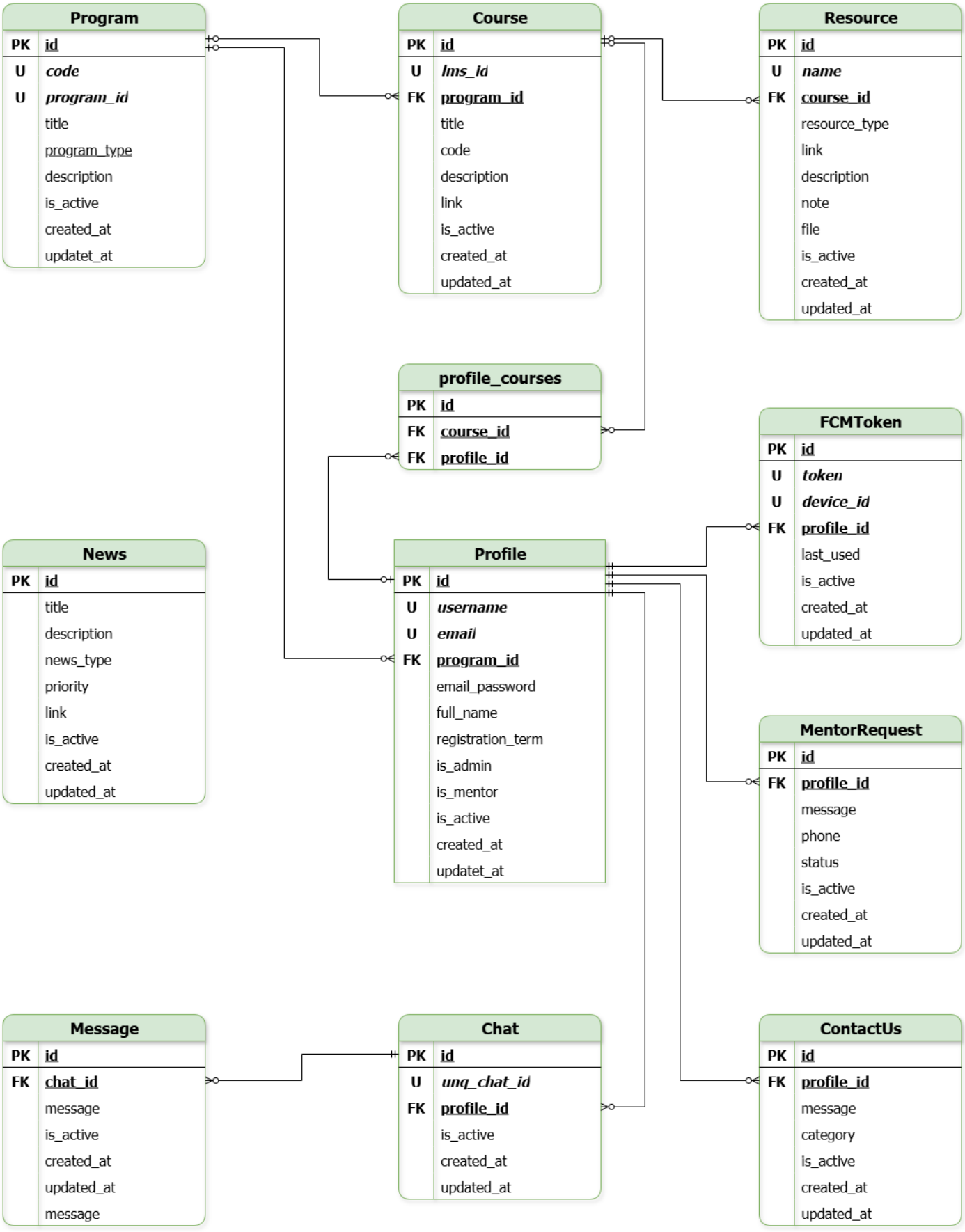
وبالتالي التصميم الجيد لقاعدة البيانات يعتبر الخطوة الأولى في عملية بناء تطبيق او نظام برمجي فعال وكفؤ يضاف الى ذلك طبعا في مرحلة التنفيذ ضرورة تطبيق بعض التقنيات مثل الفهرسة واستعلامات البحث المحسنة الامر الذي يؤمن تحسينات في الأداء تساعد في ضمان تجربة مستخدم سلسة وفعالة.

2.1.3- مخطط الكيانات العلاقات (ERD):

قمنا برسم تصميم لقاعدة البيانات باستخدام برنامج (EDRAW) ومن خلاله تم توضيح بنية قاعدة البيانات المطلوبة من جداول وتوضيح العلاقات الضرورية بين الكيانات ونوع العلاقات مع توضيح الحقول المكونة لكل جدول ممثل لهذه الكيانات.

في كل جدول قمنا بتحديد السمات المطلوبة في كل كيان بالإضافة لتمييز المفاتيح الفريدة والمفاتيح الخارجية الممثلة للعلاقات في كل كيان مع تحديد السمات التي تستقبل فقط القيم الفريدة.

كل التفاصيل في الرسم الموجود في الشكل التالي (الشكل 3-1)



الشكل5 مخطط الكيانات العلاقات

2.1.3- تصميم الجداول في قاعدة البيانات (Database Tables):

1. جدول البرنامج Program Table:

يحتوي هذا الجدول على معلومات البرامج الأكاديمية المتاحة في النظام ويشكل المرجعية الرئيسية لربط الكيانات الأخرى ببعضها مثل المواد والطلاب.

يتضمن حقول لتخزين رمز البرنامج المختصر (MCS) على سبيل المثال، معرف البرنامج، العنوان الكامل للبرنامج، نوعه والذي قد يأخذ أحد القيم التالية حاليا (ماستر، بكالوريوس، معهد) بما يتوافق مع البرامج الموجودة في الجامعة بالإضافة لوصف مختصر عن البرنامج.

Table 20: Databas Tables - Program

Program Table			
Column	***Description***	***Data Type***	***Flags***
id	المعرف الأساسي	INT	**PK**
code	رمز البرنامج	VARCHAR(10)	**UNIQUE**
program_id	معرف البرنامج	VARCHAR(10)	**UNIQUE**
title	عنوان البرنامج	VARCHAR(100)	
program_type	نوع البرنامج	VARCHAR(20)	
description	وصف البرنامج	TEXT	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

2. جدول المادة Course Table:

يخزن هذا الجدول التفاصيل الكاملة للمقررات الدراسية المرتبطة بالبرامج الأكاديمية، ليعمل كجسر بين البرامج والموارد التعليمية.

يحتوي على معرف المقرر على (LMS)، عنوان المقرر، رمزه المختصر، البرنامج التابع له كمفتاح خارجي الى جدول البرامج، وصف مختصر عن المقرر، ورابط صفحة المقرر على نظام إدارة التعلم الخاص بالجامعة.

Table 21: Database Tables - Course

Course Table			
Column	***Description***	***Data Type***	***Flags***
id	المعرف الأساسي	INT	**PK**
lms_id	معرفLMS	VARCHAR(50)	**UNIQUE**
title	عنوان المادة	VARCHAR(255)	
program	البرنامج	INT	**FK → Program**
code	رمز المادة	VARCHAR(10)	
description	وصف المادة	TEXT	
link	رابط المادة	URL	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

3. جدول المصادر Resource Table:

يحتوي هذا الجدول على المصادر التعليمية الاضافية التي يقوم المرشدين بإضافتها ويدعم أنواع مصادر متعددة (مقالات، فيديوهات، ملفات، روابط اجتماعية) المرتبطة بالمواد الدراسية.

يؤمن تصميم الجدول التحقق من عدة شروط باستخدام وظائف للتحقق من حجم ونوع الملفات ووظائف تساعد على التخزين الذكي للملفات والتوليد الالي للمسارات.

Table 22: Databas Tables - Resourse

Column	*Description*	*Data Type*	***Flags***
id	المعرف الأساسي	INT	**PK**
name	اسم الملف	VARCHAR(255)	
link	رابط خارجي	URL	
description	الوصف	TEXT	
note	ملاحظة	VARCHAR(255)	
resource_type	نوع الملف	VARCHAR(20)	
course	المادة	INT	**FK → Course**
file	الملف	FILE	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

4. جدول الملف الشخصي Profile Table:

يعتبر جدول الملف الشخصي بمثابة العمود الفقري لنظام إدارة المستخدمين والذي يقوم بتخزين كل المعلومات المتعلقة بالطالب بما في ذلك الاسم الكامل، اسم المستخدم، البريد الالكتروني، البرنامج الذي قام الطالب بالتسجيل فيه، فصل التسجيل، بيانات التسجيل، كلمة مرور البريد الإلكتروني المشفرة، والصلاحيات المعطاة له في هذا النظام والتي تحدد الدور الذي يقوم به.

Table23 : Database Tables - Profile

Column	*Description*	*Data Type*	***Flags***
id	المعرف الأساسي	INT	**PK**
username	اسم المستخدم	VARCHAR(50)	**UNIQUE**
full_name	الاسم الكامل	VARCHAR(255)	
program	البرنامج المسجل فيه (مفتاح خارجي)	INT	**FK → Program**
registration_term	فصل التسجيل	VARCHAR(10)	
email	البريد الإلكتروني	EMAIL	**UNIQUE**
email_password	كلمة مرور مشفرة	EncryptedChar(255)	
courses	المواد المسجلة	INT	**M2M → Course**
is_mentor	حالة المرشد	BOOLEAN	
is_admin	حالة المدير	BOOLEAN	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

ونظرا لأهمية هذا الجدول سنقوم بالحديث عنه بشيء من التفصيل

الحقول الرئيسية:

- المعرف الرئيسي (id): وهو رقم معرف فريد يتم إنشاؤه لكل ملف جديد.
- اسم المستخدم (username): وهو أيضا حقل فريد يحتوي اسم المستخدم كما هو على نظام المعلومات الخاص بالجامعة.
- الاسم الكامل (full_name): اسم الطالب الكامل كما يظهر في "SVU IS".
- البريد الالكتروني (email): يحوي عنوان البريد الالكتروني للطالب الخاص بالجامعة.
- كلمة مرور البريد الالكتروني المشفرة (email_password): حيث يتم تخزين كلمة مرور البريد الالكتروني بشكل مشفر وآمن لاستخدامها لاحقا في خدمة البريد الالكتروني.
- البرنامج (program): وهو مفتاح خارجي الى جدول البرامج يخزن معرف البرنامج الأكاديمي الذي يدرسه الطالب في علاقة متعدد الى واحد (many to one) أي المستخدم ينتمي الى برنامج أكاديمي واحد.
- فصل التسجيل أو الدفعة (registration_term): يخزن رقم دفعة الطالب او فصل التسجيل.
- المواد المسجلة (courses): هذا الحقل يرتبط من خلاه جدول الملف الشخصي بجدول المواد بعلاقة (Many to Many) حيث يمكن لطالب ان يسجل في أكثر من مقرر والمقرر الواحد يرتبط أيضا بالعديد من الطلاب.
- حالة المدير (is_admin): وهي تخزن قيمة منطقية (صح، خطأ) تحدد هل هذا الحساب مدير أم لا.
- حالة المرشد (is_mentor): وهي تخزن أيضا قيمة منطقية (صح، خطأ) تحدد فيما إذا كان هذا الحساب مرشد أم لا.

تحسينات الأداء

تم تصميم هذا الجدول ليوفر أداء ممتازاً مع الحفاظ على سلامة البيانات وتم تطبيق الفهرسة المتقدمة باستخدام عدة فهارس لتسريع الاستعلامات بالإضافة لتحسين المساحة باختيار أنواع البيانات المناسبة لكل حقل مع القيود المناسبة.

5. جدول رموز المصادقة FCMToken Table:

يخزن هذا الجدول رموز FCM للأجهزة التي يسجل منها المستخدم الدخول حيث يدعم النظام الدخول المتعدد من أكثر من جهاز.

يحتوي على مفتاح خارجي للملف الخاص بالمستخدم والرموز المعرفة الفريدة الخاصة بخدمة الاشعارات ومعرف فريد للجهاز ويتم استخدام هذه المعرفات في تنظيم عمليات تسجيل الدخول والمصادقة والتخويل حيث يتم التخويل للطلبات من أي جهاز وفقا لمعرفه ومعرف خدمة الاشعارات الفريدين مع تتبع آخر استخدام.

Table24 : Database Tables - FCMToken

Column	*Description*	*Data Type*	***Flags***
id	المعرف الأساسي	INT	**PK**
profile	المستخدم (مفتاح خارجي الى جدول الحسابات)	INT	**FK → Profile**
token	رمز FCM	VARCHAR(255)	**UNIQUE**
device_id	معرف الجهاز	VARCHAR(255)	
last_used	آخر استخدام	DATETIME	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

6. جدول طلب الارشاد Mentor Request Table:

ينظم هذا الجدول عملية تقديم طلبات الارشاد وتخزين الحالة الحالية للمرشد مع التفاصيل، حيث يحتوي على تفاصيل طلبات المستخدمين ليكونوا مرشدين، بما فيها الملف الشخصي، محتوى الرسالة ورقم هاتف للتواصل بالإضافة الى حالة الطلب.

Table25 : Database Tables - Mentor Request

Column	Description	Data Type	Flags
id	المعرف الأساسي	INT	**PK**
profile	المستخدم (مفتاح خارجي الى جدول الحسابات)	INT	**FK → Profile**
message	الرسالة	TEXT	
phone	رقم الهاتف	VARCHAR(15)	
status	حالة الطلب	BOOLEAN	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

7. جدول التواصل ContactUs Table:

يخزن هذا الجدول رسائل المستخدمين المرسلة عبر نموذج الاتصال، ويحدد الملف الشخصي للمرسل، محتوى الرسالة وتصنيف الرسالة ولدينا حاليا التصنيفات التالية (مشكلة – طلب ميزة – استفسار).

يعمل هذا الجدول كقناة اتصال رسمية بين المستخدمين والإدارة.

Table26 : Database Tables - ContactUs

Column	Description	Data Type	Flags
id	المعرف الأساسي	INT	**PK**
profile	المستخدم (مفتاح خارجي الى جدول الحسابات)	INT	**FK → Profile**
message	محتوى الرسالة	TEXT	
category	التصنيف	VARCHAR(20)	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

8. جدول الأخبار News Table:

يحتوي هذا الجدول على الاخبار ويحتوي عنوان الخبر، والمحتوى وهو موجز عن الخبر الرئيسي، نوع الخبر او تصنيفه حيث لدينا حاليا ثلاث تصنيفات هي الأخبار الخاصة بالجامعة – الاخبار المتعلقة بالتعليم وتصنيف للأخبار العامة، بالإضافة الى وجود مستوى للأهمية أو الأولوية منخفض– متوسط وعال يمكن استخدامه لتخصيص الاشعارات وأخيرا رابط الخبر الأصلي للحصول على التفاصيل الكاملة عند الضرورة.

Table27 : Database Tables - News

Column	Description	Data Type	Flags
id	المعرف الأساسي	INT	**PK**
title	عنوان الخبر	VARCHAR(255)	
description	محتوى الخبر (موجز عن الخبر الرئيسي)	TEXT	
link	رابط الخبر	URL	

news_type	نوع الخبر	VARCHAR(20)	
priority	الأولوية	VARCHAR(20)	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

9. جدول المحادثات Chat Table:

يحتوي هذا الجدول على المحادثات وفيه نجد الملف الشخصي للطالب صاحب المحادثة، معرف فريد للمحادثة من التطبيق، رقم الهاتف، وحالة الطلب.

Table28 : Database Tables - Chat

Column	*Description*	*Data Type*	***Flags***
id	المعرف الأساسي	INT	**PK**
profile	المستخدم (مفتاح خارجي الى جدول الحسابات)	INT	**FK → Profile**
unq_chat_id	معرف فريد للمحادثة	VARCHAR(255)	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

10. جدول رسائل المحادثات Message:

يحتوي هذا الجدول على رسائل المحادثات المرتبطة بمحادثة محددة، نوع الرسالة هل هي سؤال ام إجابة، محتوى الرسالة والطابع الزمني للرسالة كما وردت ضمن المحادثة.

Table29 : Database Tables - Message

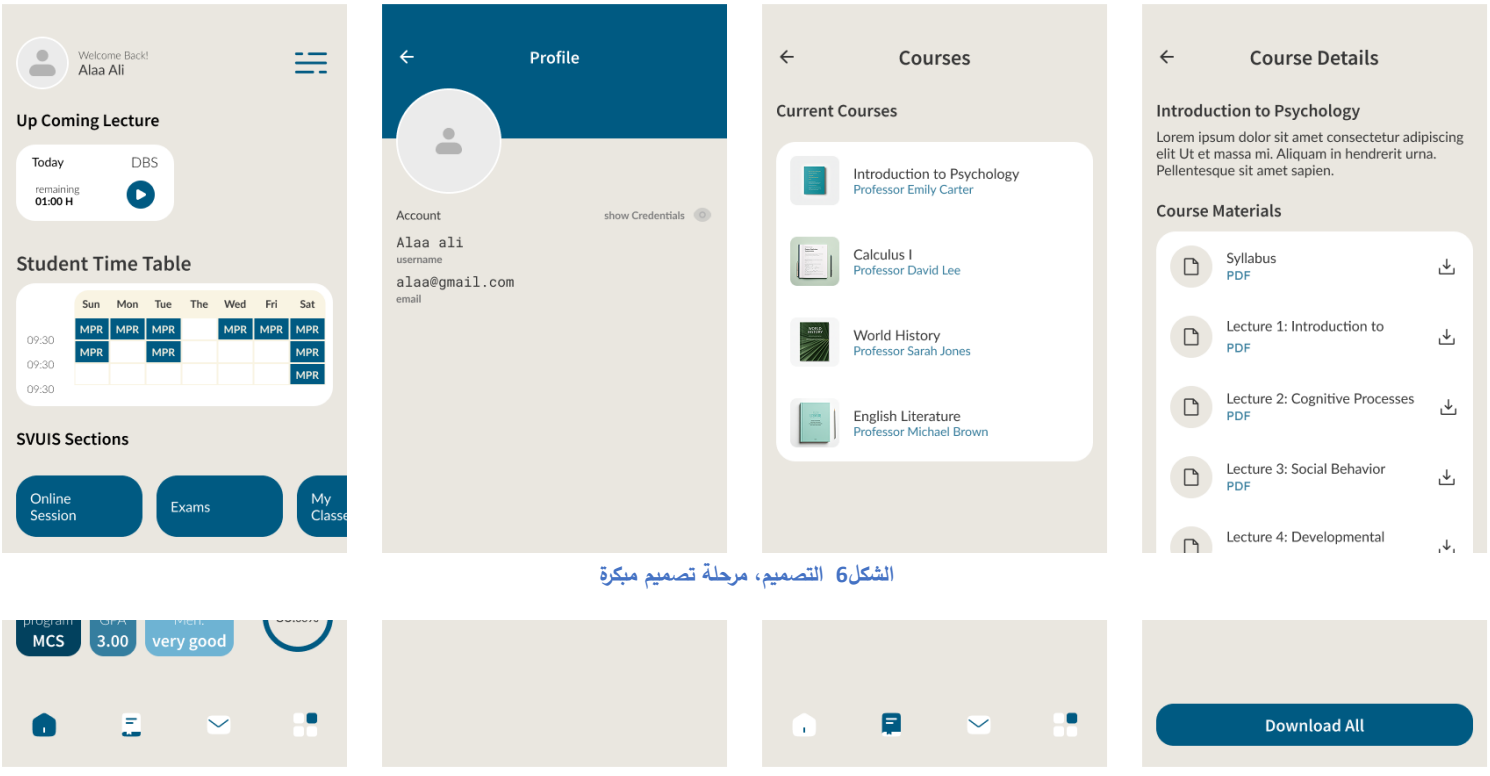
Column	Description	Data Type	Flags
id	المعرف الأساسي	INT	**PK**
chat	مفتاح خارجي الى جدول المحادثات	INT	**FK → Profile**
message_type	نوع الرسالة (سؤال – إجابة)	VARCHAR(255)	
content	محتوى الرسالة	TEXT	
timestamp	طابع زمني لوقت الرسالة ضمن المحادثة	DATETIME	
created_at	تاريخ الانشاء	DATETIME	
updated_at	تاريخ التعديل	DATETIME	
is_active	الحالة نشط/غير نشط	BOOLEAN	

2.3- التصميم المرئي (UI/UX Design):

كوننا اعتمدنا على نموذج النموذج الاولي فقد قمنا في البداية بوضع رسومات بسيطة وسريعة على الورق لتكوين تصور اولي بسيط عن الواجهات المطلوبة وتوزع الوظائف فيها.

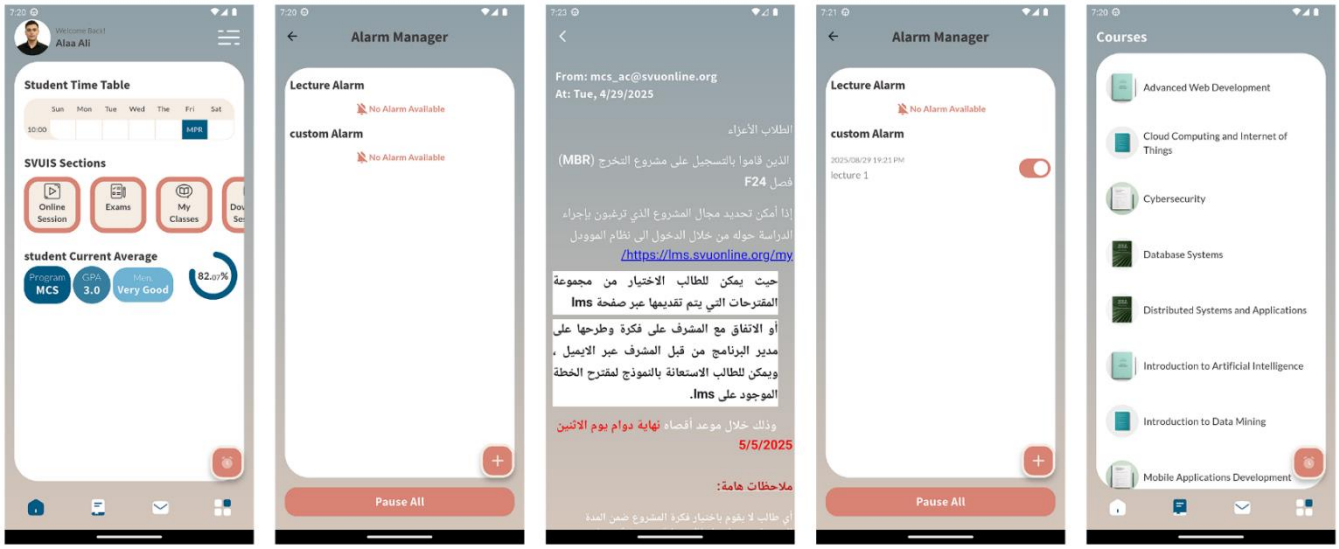
ثم تم الانتقال الى التصميم باستخدام تطبيق (Figma) وتم التطوير بالتصميم على عدة مراحل مع التقدم بالعمل ومع كل نموذج بالتواصل مع بعض المستخدمين المتطوعين لإبداء آرائهم في التصميم وبأخذ ملاحظاتهم تم تطوير التصميم بشكل متواصل حتى الوصول الى التصميم النهائي.

في الشكل التالي (الشكل 6) تظهر صورة من أحد مراحل التصميم باستخدام (Figma) في المراحل الأولى لتكوين نواة التصميم وتظهر فيه المراحل الأولية للتصميم والتي تم تغيير العديد من مكوناته وإعادة ترتيبها كما تم تغيير الألوان واشكال الازرار وتصميم المحتوى ونوع الخطوط وبعملية مستمرة ومتتالية.



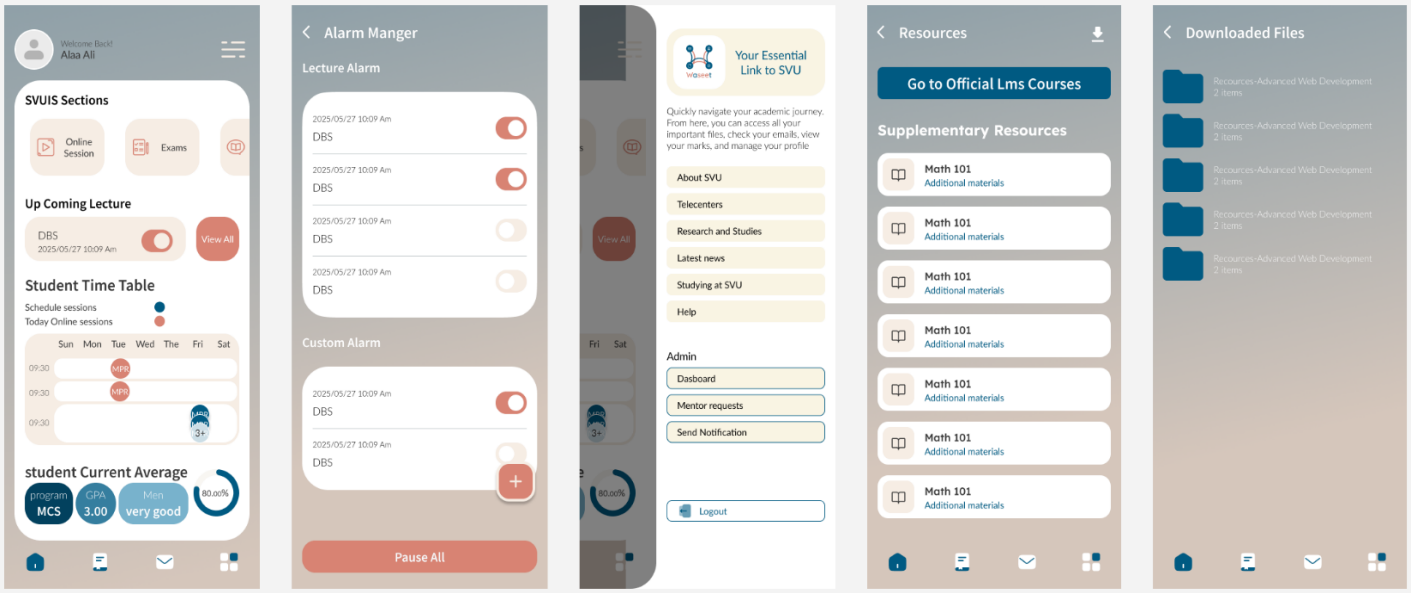
الشكل6 التصميم، مرحلة تصميم مبكرة

وفي الشكل التالي تظهر احدى مراحل التصميم في مرحلة وسطية من التطوير وتظهر بعض التغييرات على التصميم عن الواجهات السابقة



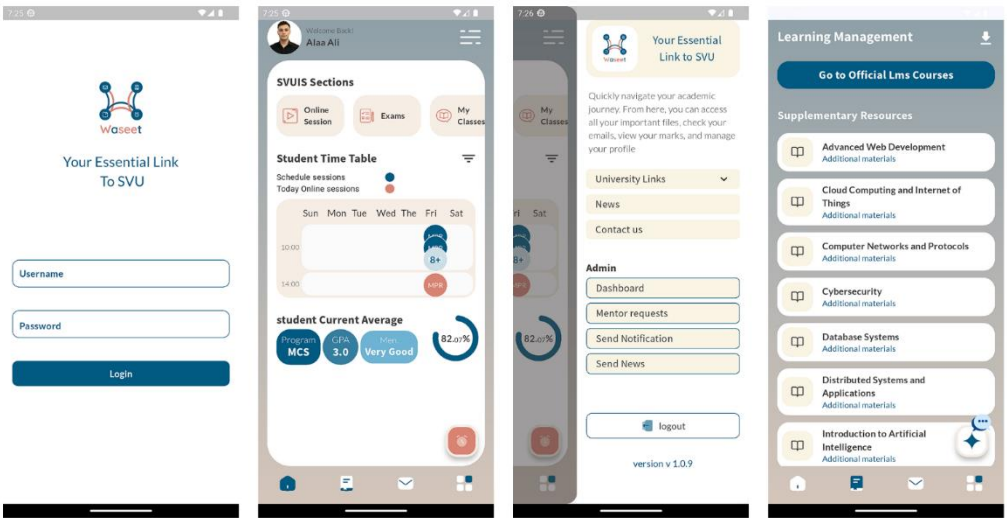
الشكل7 التصميم، مرحلة تصميم وسطية

وفي الشكل التالي احدى مراحل التصميم المتقدمة وتظهر فيها بعض التغييرات

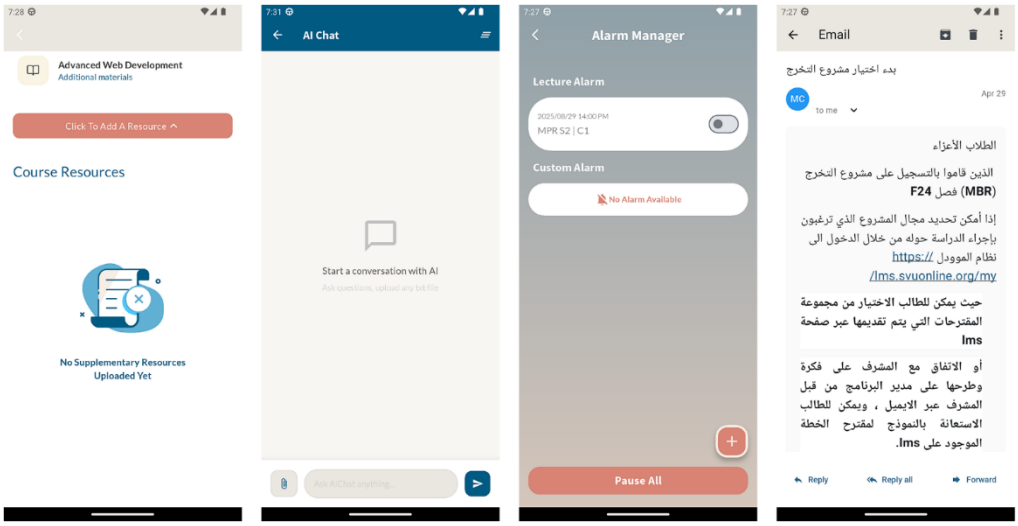


الشكل8 التصميم، مرحلة تصميم متقدمة

وكما ذكر مسبقا بالاعتماد على التغذية الراجعة من بعض المستخدمين وبإعادة النظر ببعض التفاصيل ومدى توافقها مع معايير تجربة المستخدم تم الوصول الى المرحلة التالية ضمن تطور عملية التصميم والتي تظهر في الشكل التالي ()



الشكل 9 التصميم، مراحل التصميم النهائي 1



الشكل 10 التصميم، مراحل التصميم النهائي 2

يظهر في الصور السابقة بوضوح مجموعة من التعديلات على التصميم شملت الألوان ومواقع المكونات وتصميم الاشكال.

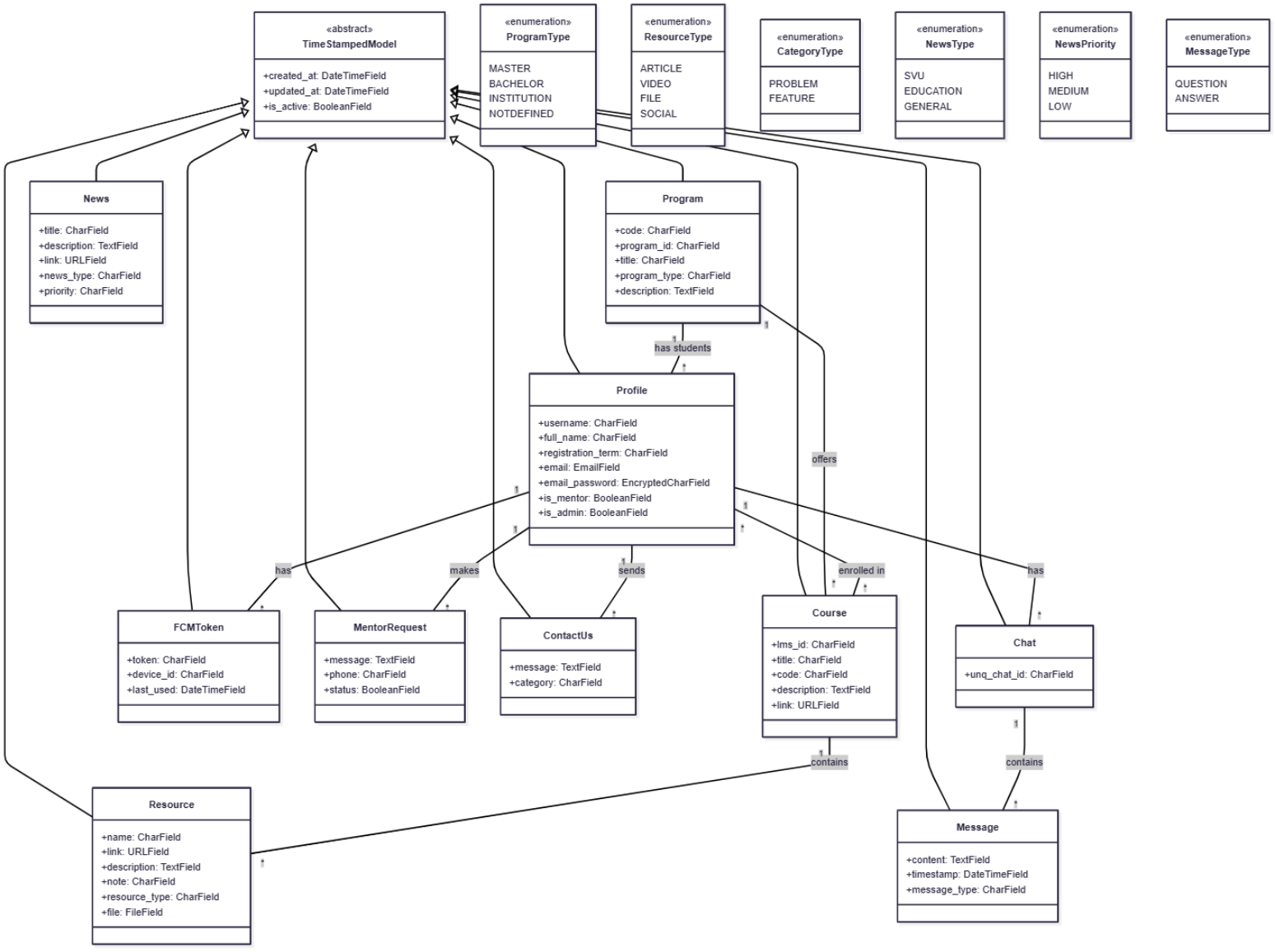
ومع تقدم العمل استمرت عملية تحسين التصميم بشكل مستمر حتى الوصول الى التصميم النهائي للتطبيق، وسيتم في الفصل الخاص بالاختبار عرض لقطات شاشة من عمل التطبيق في تصميمه النهائي.

3.3- مخططات الفئات (Class Diagrams):

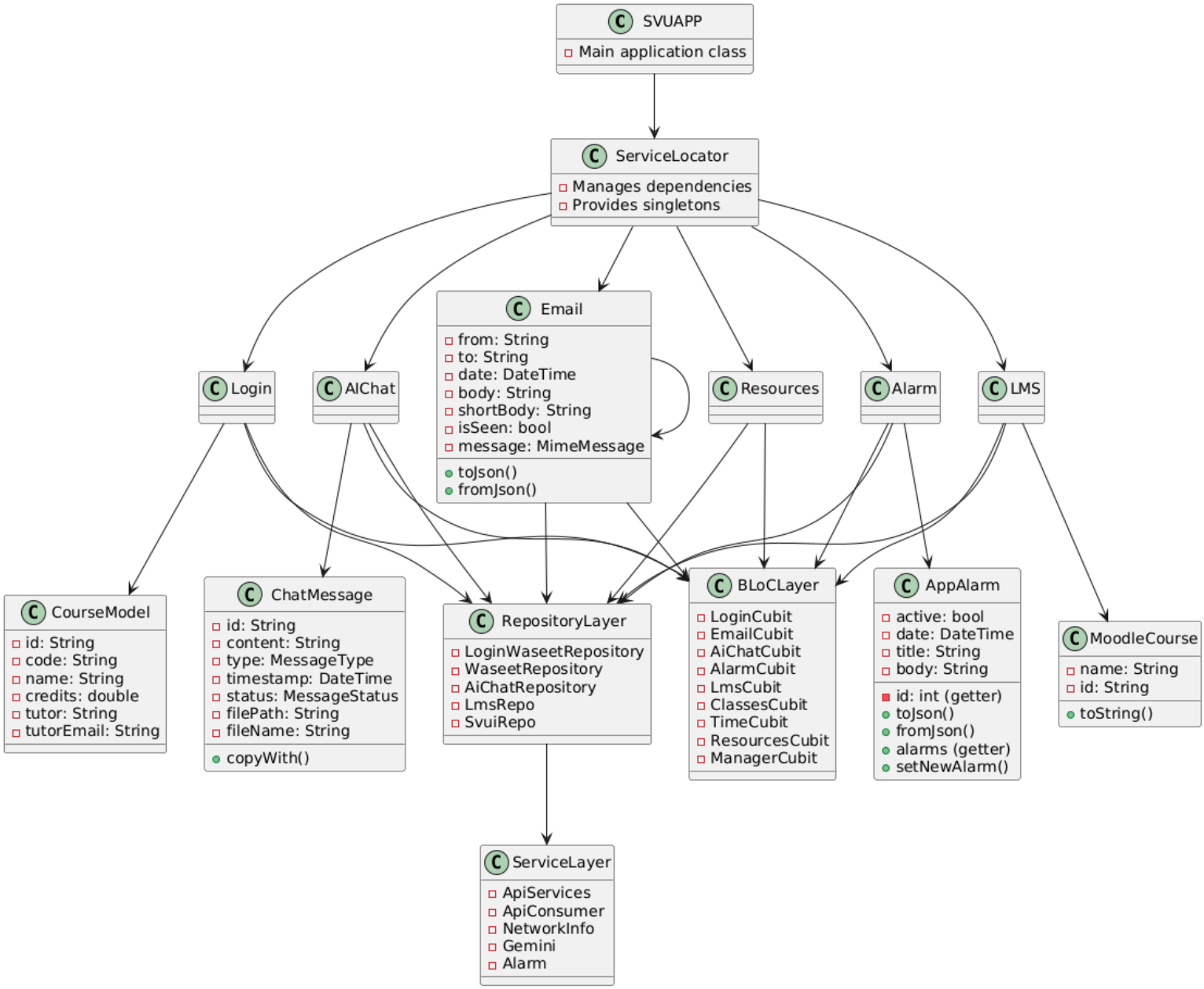
تعد مخططات الفئات أحد أهم المخططات الهيكلية في لغة النمذجة الموحدة حيث تقدم تصوراً لبناء الأنظمة الموجهة للكائنات (Object Oriented Systems) وهو بمثابة مخطط معماري يوضح هيكل النظام البرمجي مما يساعد على فهم كيفية تنظيم الأجزاء المختلفة للنظام وتفاعلها.

يظهر هذا المخطط بشكل أساسي الفئات (Classes), السمات (Attributes)، العمليات والدوال (Operations/Methods) بالإضافة أيضا للعلاقات (Relationships) مثل الارتباط "Association"، الوراثة والتعميم "Inheritance/Generalization" والتجميع "Aggregation" وغيرها.

في الشكلين التاليين نقدم مخطط الفئات لكل من تطبيق الفلاتر ولخدمات الخلفية



الشكل11 مخطط الفئات للواجهة الخلفية



الشكل12 مخطط الفئات للواجهة الأمامية

الفصل الرابع: التنفيذ

1.4- مقدمة:

بعد الانتهاء من مرحلتي التحليل والتصميم يتم الانتقال الى مرحلة التنفيذ (Implementation) والتي تعرف أحيانا بمرحلة التطوير (Development) أو كتابة الكود (Coding)

في هذه المرحلة يتم تحويل الأفكار والمتطلبات التي جرى تحليلها والتصاميم المعمارية الى واقع عملي بكتابة الكود المصدري الذي سينفذها.

يتم البدء بكتابة الأوامر والجمل البرمجية لبناء المكونات من وحدات وبرامج فرعية بما يتوافق مع المخططات والتصميم ثم يتم تجزيع الأجزاء المبرمجة معا لتكوين النظام النهائي.

وبحكم اننا نقوم بتطوير واجهة أمامية باستخدام (Flutter) وواجهة خلفية باستخدام (Django) والربط بينهما بواجهات التخاطب البرمجية (APIs) فيجب الانتقال الى كل منها والبدء بكتابة الكود المصدري المناسب.

2.4- تنفيذ الواجهة الخلفية وواجهات التخاطب البرمجية:

كما ذكرنا سابقا تم الاعتماد على إطار العمل جانغو (Django) لتحقيق خدمات الخلفية ومن خلاله أيضا وبالاعتماد على الإطار (Django Rest Framework) تم بناء واجهات التخاطب البرمجية.

البداية ستكون من جانغو حيث سنقوم بإنشاء مشروع ونختار له اسم (mcs-backend)

بعد التأكد من تثبيت لغة البرمجة بايثون بالاصدارة المطلوبة "3.10.11" على الجهاز ننشئ مجلد جديد (alaa) ونفتح بداخله موجه الأوامر (CMD) ونقوم بالخطوات التالية:

- تثبيت مكتبة البيئات الافتراضية بالأمر "*pip install virtualenv*"
- ننشئ بيئة افتراضية خاصة للمشروع (venv310) لتجنب حدوث أي تعارضات أثناء التطوير مع التأكيد على استخدام اصدارة بايثون المطلوبة وذلك بالأمر التالي "*virtualenv -p python3.10 venv310*".
- تفعيل البيئة الافتراضية التي تم انشاؤها والتي سيتم تثبيت كافة الاعتماديات فيها وذلك باستخدام الامر "*venv310/Scripts/activate*".

- نقوم بتثبيت الاعتماديات والمكتبات المطلوبة دفعة واحدة باستخدام الأمر "*pip install -r requirements.txt*".
- ثم الان نقوم بإنشاء المشروع بالاسم المطلوب وذلك باستخدام الأمر "*django-admin startproject mcs-backend*".
- سنستخدم تطبيق واحد في هذا المشروع لإدارة خدمات الخلفية وليكن اسمه (backend) ونقوم بإنشائه بالأمر التالي "*django-admin startapp backend*".

الان بعد ان تم انشاء المشروع وتطبيق خدمات الخلفية بداخله نبدأ بعملية البرمجة والتي تشمل في حالتنا بداية كتابة الاكواد البرمجية الأساسية في ملفات جانغو الرئيسية بما يتوافق مع التصميم المقترح.

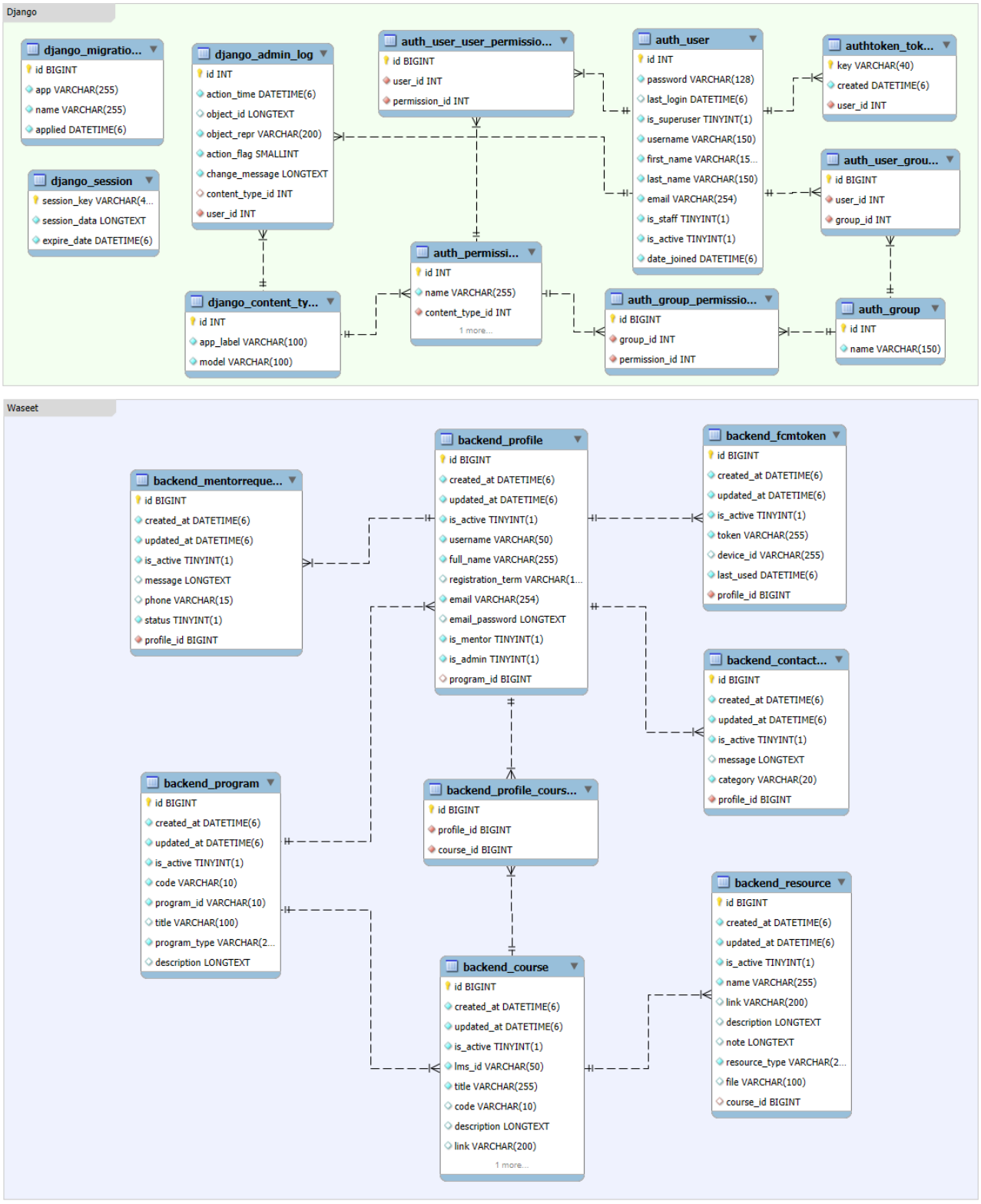
1.2.4- تنفيذ قاعدة البيانات:

البداية ستكون بتنفيذ قاعدة البيانات والتي كما ذكرنا سابقا ستكون من النوع SQLite بشكل افتراضي ولن نقوم حاليا بتغييرها ضمن مرحلتي التطوير والاختبار.

في جانغو يتم تنفيذ قاعدة البيانات بتعريف الصفوف ضمن ملف (models.py) وفقا للتصميم السابق وبعدها يقوم جانغو بإدارة عمليات قاعدة البيانات عن طريق واجهة تخاطب داخلية مع قاعدة البيانات هي رابط الكائن العلاقة (ORM).

وبعد الانتهاء من تعريف الصفوف يجب تطبيق هذه التعريفات على قاعدة البيانات ولذلك نستخدم أولا الامر "*manage.py makemigrations*" لإخبار جانغو بوجوب البحث عن وجود تغييرات في تعريف قاعدة البيانات وانشاء التهجيرات الموافقة.

ثم لتطبيق هذه التغييرات نستخدم الأمر "*manage.py migrate*" والذي يقوم بتحويل هذه التغييرات الى تعليمات SQL وتنفيذها على مستوى قاعدة البيانات وبالتالي يتم انشاء الجداول الموافقة لتعريفات الصفوف والمبينة في الشكل التالي:



الشكل13 مخطط قاعدة البيانات بعد التنفيذ

2.2.4- تنفيذ خدمات الواجهة الخلفية وواجهات التخاطب البرمجية:

تقوم الواجهة الخلفية والتي يُطلق عليها أحيانًا اسم جانب الخادم بإدارة الوظائف العامة للتطبيق، ومع تفاعل المستخدم مع الواجهة الأمامية والتي هي في حالتنا تطبيق الجوال يتم ارسال طلب الى الواجهة الخلفية بتنسيق (HTTP) عبر واجهات برمجة التطبيقات (API) المعرفة في التطبيق لتقوم بعدها الواجهة الخلفية بتلقي الطلب ومعالجته ثم إرجاع الاستجابة المناسبة ليتم عرضها في التطبيق.

في مشروعنا الحالي لدينا في الواجهة الخلفية تطبيق خدمات الخلفية (backend) والذي يدير الوظائف المطلوبة وعمليات المعالجة وذلك بالاعتماد على الشيفرة البرمجية المكتوبة في وحدات مخصصة هي عبارة عن ملفات بايثون يتم انشاؤها عند انشاء التطبيق ومن ثم نقوم نحن بكتابة الكود المطلوب فيها والمناسب لحالات الاستخدام وللتصميم المقترح.

كل ملف من هذه الملفات يقوم بوظيفة او مجموعة من الوظائف المحددة، هذه الملفات هي:

- ***urls.py***

يحتوي على مجموعة عناوين نقوم بتعريفها ويتم ربط كل عنوان مع دالة واحدة حيث يتم توجيه الطلب الى العرض المناسب بناء على عنوان URL الخاص بالطلب لتقوم الدالة او الصف ضمن العرض بمعالجة الطلب وارسال الاستجابة المناسبة حسب الشيفرة البرمجية، وفيه نضع العناوين لصفحات الموقع أو كما في حالتنا عناوين نقاط النهاية (endpoints) التي سنرسل لها الطلبات من التطبيق.

- ***models.py***

يحتوي نماذج البيانات على شكل صفوف تمثل بنية بيانات التطبيق وقد تعرفنا عليها في الفقرة السابقة عند تنفيذ قاعدة البيانات.

- ***views.py***

من اهم الملفات وهو العقل والمنطق في التطبيق ويحتوي على العروض والتي هي دوال او صفوف تقوم بمعالجة الطلبات حيث تتلقى طلب HTTP وتعيد استجابة HTTP وتحصل على البيانات المطلوبة لهذه الاستجابة عن طريق النماذج (Models) وعند الضرورة تقوم القوالب (Templates) بتنسيق هذه الاستجابة.

ومن ضمن هذه الدوال مجموعة الدوال الخاصة بواجهات برمجة التطبيقات (API) والتي تتلقى طلب من التطبيق في الواجهة الامامية لتعيد استجابة بيانات او معالجة يتم عرضها في التطبيق

سنعتمد بشكل كبير على مكتبة Django Rest في تعريف دوال هذه الواجهة.

- ***serializers.py***

في هذا الملف يتم تعريف صفوف السلسلة والتي تقوم بسلسلة البيانات بتنسيق مناسب وهنا نستخدم تنسيق JSON حيث نعرف لكل نموذج من النماذج المعرفة لدينا في الملف (Models) صف سلسلة أو أكثر خاص به لنستخدمه في معالجة البيانات في الطلب والاستجابة.

- ***admin.py***

في هذا الملف نقوم بضبط اعدادات واجهات الإدارة (Admin Dashboard) وتخصيصها بالشكل المناسب لاحتياجاتنا حيث سنستخدم واجهة الإدارة في ادخال البيانات الأساسية في النظام ولاحقا في عمليات الإدارة.

يمكننا هذا الملف بالتحكم بالكثير من الاعدادات الخاصة بواجهة الإدارة مثل الاعمدة التي ستعرض عند عرض البيانات الخاصة بالنموذج وطريقة عرضها مثلا تجميع بعض البيانات المترابطة في مجموعات والتحكم بصلاحيات القراءة والكتابة كما يمكننا إضافة فلاتر لفلترة هذه البيانات.

مجموعة ملفات مساعدة تحوي وظائف إضافية مخصصة:

- ***authentication.py***

في هذا الملف نقوم بتعريف صف التخويل الخاص بنا على السيرفر والذي سنعتمد فيه على المعرف الفريد (FCM Token) كون عملية التخويل لخدمات الجامعة تتم على سيرفرات الجامعة بالاعتماد على اسم المستخدم وكلمة المرور.

- ***helper.py***

في هذا الملف نقوم بتعريف مجموعة من الوظائف المساعدة التي نحتاجها مثل الدالة (resource_upload_path) التي تقوم بإنشاء المسار الخاص بتحميل الملفات بشكل ديناميكي.

- ***validators.py***

في هذا الملف نقوم بتعريف مجموعات التحقق (Validators) الخاصة بنا مثل (validate_file_size – validate_file_extension) والتي تقوم بالتحقق من اسم الملف وحجمه بشكل مسبق قبل رفعه وإصدار التحذيرات المناسبة عند عدم تحقق الشروط.

- ***notification.py***

في هذا الملف نقوم بتعريف الدوال الخاصة بإرسال الاشعارات في عدة حالات مثل (send_news_notifications – send_notification_to_students) والتي تقوم بمعالجة عملية ارسال الاشعار والتخاطب مع خدمة جوجل فيربيس لدفع الاشعارات الى أجهزة المستخدمين.

3.4- تنفيذ الواجهة الأمامية:

كما ذكرنا سابقا تم تنفيذ الواجهة الأمامية بالاعتماد على إطار العمل فلاتر ويجب توافر المتطلبات الأساسية التالية قبل البدء:

- تثبيت (Flutter SDK) على الجهاز.
- تثبيت محرر أكواد وهنا سنستخدم المحرر الشهير (Visual Studio Code).
- تثبيت إضافات (Flutter, Dart) في المحرر.

في نفس المجلد الخاص بملفات المشروع (alaa) نفتح موجه الأوامر (CMD) ونكتب فيه الأمر التالي "*flutter create waseet*"

سيقوم هذا الامر بانشاء مشروع فلاتر جديد باسم "*waseet*" وانشاء مجموعة المجلدات والملفات اللازمة.

وفي مجلد (lib) يتم انشاء ملف (main.dart) وهو نقطة الدخول للتطبيق.

نبدأ بعدها بكتابة الكود البرمجي الخاص ببناء واجهات التطبيق ومعالجة حالات الاستخدام المحددة سابقاً بالإضافة الى دمج خدمات فيربيس للإشعارات.

تم الاعتماد في التنفيذ على معمارية (Clean Architecture) مع تنظيم معتمد على المميزات (Feature-based Organization) وهو نمط تصميم للبرمجيات يهدف الى فصل أجزاء التطبيق المختلفة لزيادة قابلية الصيانة والمرونة بحيث يتم تنظيم الكود في طبقات متحدة المركز

ويكون منطق الأعمال في المنتصف وغير معتمد على الطبقات الخارجية مما يسمح بتغيير التقنيات دون التأثير على جوهر التطبيق.

وهنا اعتمدنا على تنظيم معتمد على المميزات حيث قمنا بفصل المميزات وتجميع كل الملفات المتعلقة بميزة محددة ضمن مجلد واحد.

وبالنسبة لإدارة الحالة فتم الاعتماد على (BloC Pattern) باستخدام الحزمة (flutter_bloc) والمقصود هنا بالمصطلح BloC (Business Logic Component) وهو نمط تصميمي يمكن المطور من إدارة الحالة بشكل فعال ومناسب عبر التطبيق وتحقيق الفصل بين عناصر الواجهة ومنطق الاعمال.

وتم الاعتماد على تقنية حقن الاعتماديات باستخدام حزمة (GetIt) والتي تعتبر أداة قوية ومرنة لتطبيق نمط محدد الخدمات في تطبيقات فلاتر، مما يسهم في بناء كود نظيف، فعال، وقابل للصيانة.

في الشكل التالي نبين هرمية الملفات والمجلدات المتوافقة مع المعمارية ونهج التصميم المذكور سابقا:

Main Source Code Structure (lib/)

```
lib/
├── main.dart                    # Application entry point
├── service.dart                 # Service configuration
├── splash_screen.dart           # Splash screen widget
├── onboarding_screen.dart       # Onboarding screen widget
├── core/                        # Core utilities and shared components
└── Feature/                     # Feature-based modules
```

الشكل 14 بنية الملفات الرئيسية

Core Directory Structure

```
core/
├── apis/                          # API related classes
│   ├── api_consumer.dart          # Abstract API consumer
│   ├── api_services.dart          # API service implementations
│   ├── app_exception.dart         # Custom exceptions
│   ├── dio_consumer.dart          # Dio HTTP client implementation
│   ├── dio_helper.dart            # Dio helper utilities
│   ├── end_points.dart            # API endpoints constants
│   └── network_info.dart          # Network connectivity checker
├── common_widget.dart/            # Reusable UI components
│   ├── app_button.dart            # Custom button widget
│   ├── background_container.dart # Background container widget
│   ├── back_icon.dart             # Back navigation icon
│   ├── custom_transitions.dart   # Custom page transitions
│   ├── drawer_icon.dart           # Drawer menu icon
│   ├── empty_widget.dart          # Empty state widget
│   ├── error_widget.dart          # Error state widget
│   ├── password_show_button.dart # Password visibility toggle
│   ├── refresh_widget.dart        # Pull-to-refresh widget
│   ├── shimmer.dart               # Shimmer loading effect
│   ├── side_bar_item.dart         # Sidebar menu item
│   ├── switch.dart                # Custom switch widget
│   ├── text.dart                  # Custom text widget
│   └── text_field_widget.dart     # Custom text field widget
├── constant/                      # App constants
│   ├── colors.dart                # Color constants
│   ├── enums.dart                 # Enum definitions
│   ├── extension.dart             # Extension methods
│   ├── string.dart                # String constants
│   └── text_style.dart            # Text style constants
├── errors/                        # Error handling
│   ├── error_model.dart           # Error model class
│   ├── expentions.dart            # Error extensions
│   └── failure.dart               # Failure class
├── extension/                     # Dart extensions
│   └── emty_or_null.dart          # Empty/null check extensions
├── fcm/                           # Firebase Cloud Messaging
│   ├── fcm_config.dart            # FCM configuration
│   └── push_notfi.dart            # Push notification handler
├── fun/                           # Utility functions
│   └── fun.dart                   # General utility functions
├── shared_preferences/            # Local storage
│   └── shared_preferences.dart   # SharedPreferences wrapper
├── common_widget.dart             # Common widget exports
├── data.dart                      # Data layer utilities
├── navigation_key.dart            # Global navigation key
├── service_locator.dart           # Dependency injection setup
└── webview.dart                   # WebView utilities
```

الشكل 15 بنية ملفات نواة التطبيق

الميزات منفصلة في مجلدات متعددة

Feature Directory Structure

Each feature follows a clean architecture pattern with data and presentation layers:

```
Feature/
├── ai_chat/                    # AI Chat Feature
│   ├── data/                   # Data layer (repositories, models, data sources)
│   └── presentation/           # Presentation layer (UI, BLoC, pages)
├── alarm/                      # Alarm Feature
│   ├── data/
│   └── presentation/
├── apps/                       # Apps Feature
│   ├── data/
│   └── presentation/
├── email/                      # Email Feature
│   ├── data/
│   ├── presentation/
│   └── student_support/        # Student support specific email handling
├── lms/                        # Learning Management System Feature
│   ├── data/
│   └── presentation/
├── login/                      # Authentication Feature
│   ├── data/
│   └── presentation/
├── manager/                    # Manager Feature
│   ├── data/
│   └── presentation/
├── resources/                  # Resources Feature
│   ├── data/
│   └── presentation/
└── svui/                       # SVU Interface Feature
    ├── data/
    └── presentation/
```

الشكل 16 بنية الملفات والمجدات للمميزات وفق المعمارية النظيفة

4.4- الشيفرة البرمجية (Source Code):

الشيفرة البرمجية لكل من الواجهة الأمامية والخلفية تم جمعها في ملف مضغوط باسم (Source Code) ورفعها مع التقرير وبقية الملفات المطلوبة على نظام إدارة التعلم (LMS).

الفصل الخامس: النشر والاختبار

1.5- مقدمة:

بعد الانتهاء من مرحلة التنفيذ ننتقل الى مرحلتي النشر والاختبار حيث في مرحلة النشر نقوم بنقل التطبيق والنظام من بيئة التطوير الى بيئة الإنتاج حيث يمكن للمستخدمين النهائيين الوصول الى التطبيق واستخدامه بشكل سليم.

وبعد اكتمال النشر سنقوم بإجراء بعض الاختبارات الفعلية ووضع بعض الصور من الخرج والواجهات.

2.5- النشر:

تشمل عملية النشر في قسمها المتعلق بالواجهة الامامية بناء نسخة من التطبيق في إطار العمل فلاتر بحيث تكون جاهزة للتثبيت على أجهزة المستخدمين.

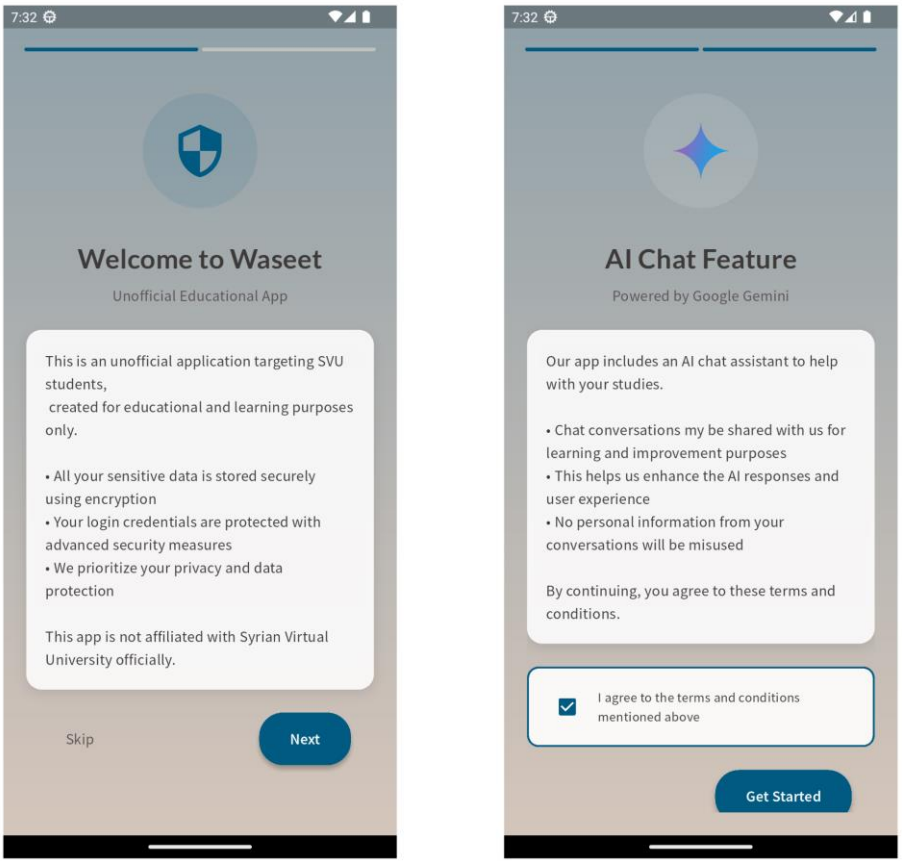
وفيما يتعلق بالواجهة الخلفية يجب اختيار استضافة مناسبة ونقل الملفات اليها مع ضبط الاعدادات المختلفة للمشروع بحسب التوثيق الخاص بالاستضافة.

سنقوم باستخدام خدمة الاستضافة المجانية المقدمة من Python Anywhere والتي تقدم خادم ويب بحجم مقبول بحدود 512 ميغابايت وهي مناسبة لمشروعنا حاليا.

ولتجهيز المشروع للنشر سنقوم بإجراء بعض التعديلات الهامة على ملف الاعدادات (Settings.py) لإخفاء بعض القيم الحساسة مثل المفتاح الرئيسي السري الخاص بإطار العمل دجانغو "SECRET_KEY " ومفاتيح التشفير "FIELD_ENCRYPTION_KEY " وغيرها بوضعها ضمن متغيرات البيئة ولتحقيق ذلك بالشكل الأمثل سنقوم باستخدام مكتبة (Python Decouple) لنقوم بتخزين القيم الحساسة ضمن ملف (.env) والذي يتم قراءته واخذ القيم منه عند بدء تشغيل السيرفر مع اسناد قيم افتراضية مناسبة في حال لم يتم تزويد ملف البيئة.

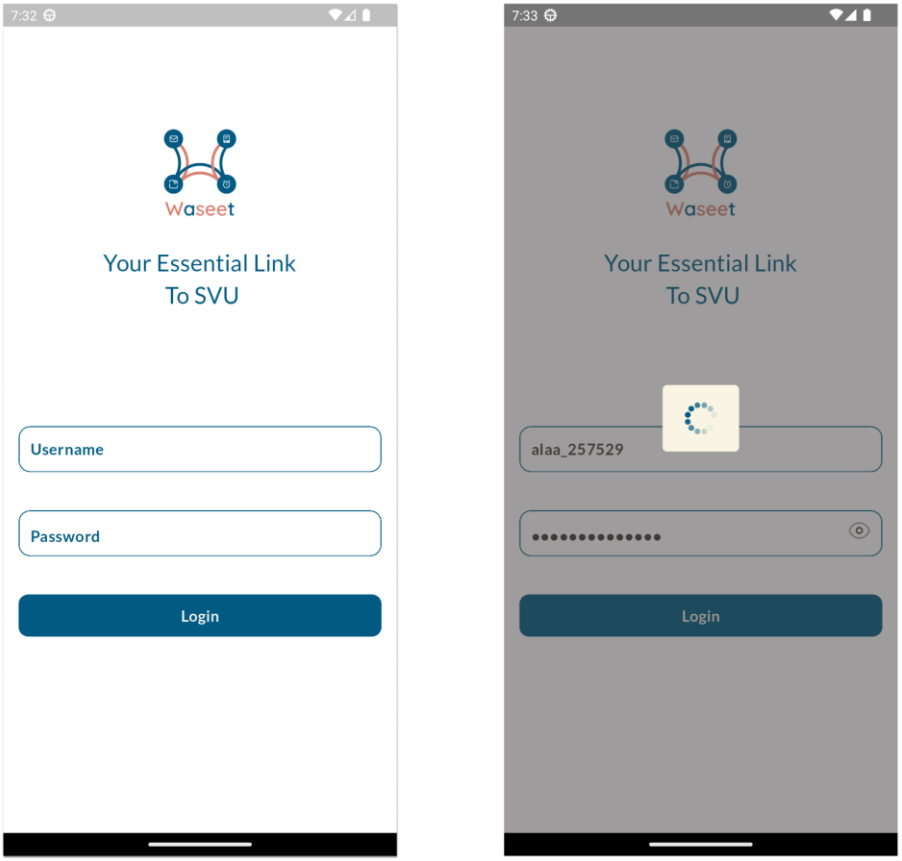
3.5- الاختبارات العملية:

سنقوم بعرض صور بعض الواجهات والتي تم التقاطها اثناء الاختبار العملي للتطبيق ونبدأ من واجهة الشروط والأحكام وهي أول واجهة تظهر فقط في أول مرة يتم فتح التطبيق فيها بعد تثبيته وتظهر في الصورة التالية



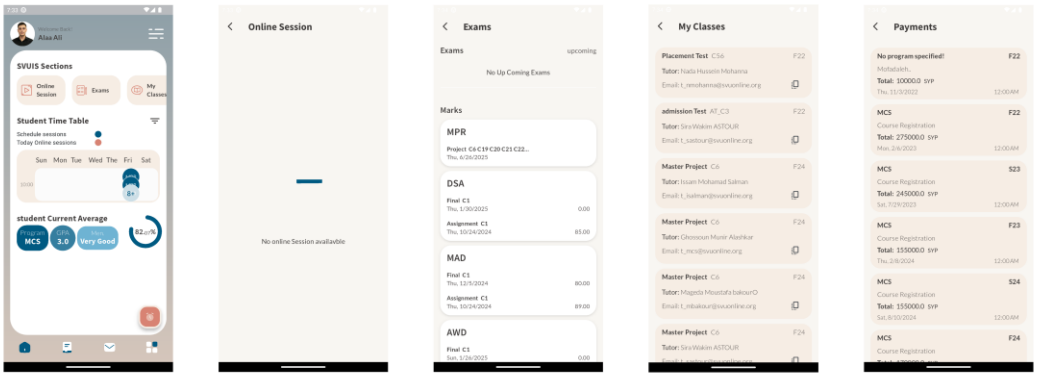
الشكل 17 – الاختبار – واجهة الشروط

الواجهة التالية والتي تظهر مباشرة بعد الموافقة على شروط استخدام التطبيق هي واجهة تسجيل الدخول والمبينة في الصور التالية

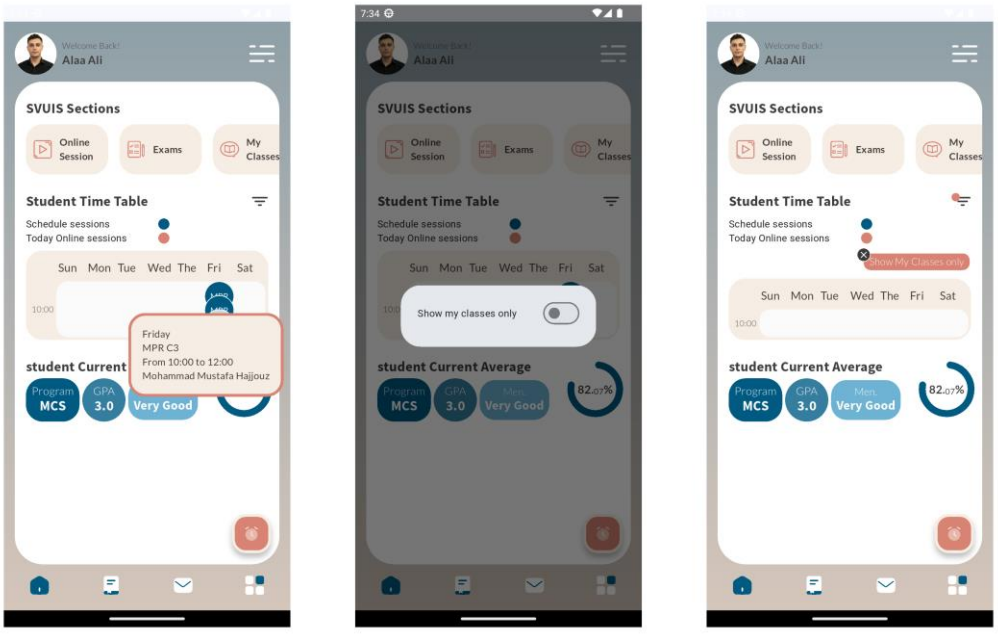


الشكل 18 – الاختبار – واجهة تسجيل الدخول

واجهات معلومات المستخدم وتظهر فيها معلومات الجلسات والامتحانات ونتائجها والصفوف وسجل الدفعات المالية وجدول المحاضرات وغيرها كما يظهر في الصور التالية



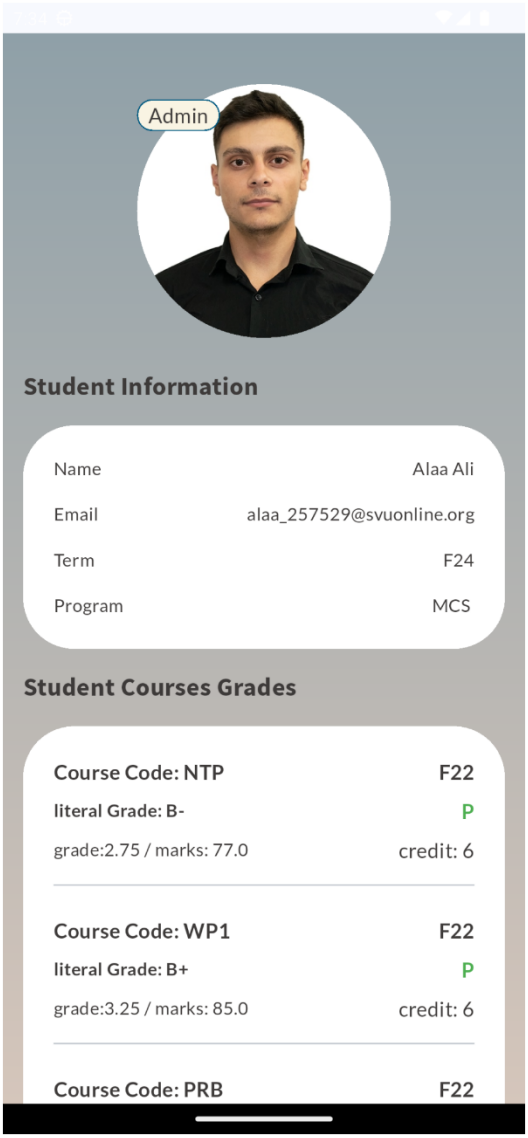
الشكل 19 – الاختبار – واجهات معلومات المستخدم 1



الشكل 20 – الاختبار – واجهات معلومات المستخدم 2

الملف الشخصي:

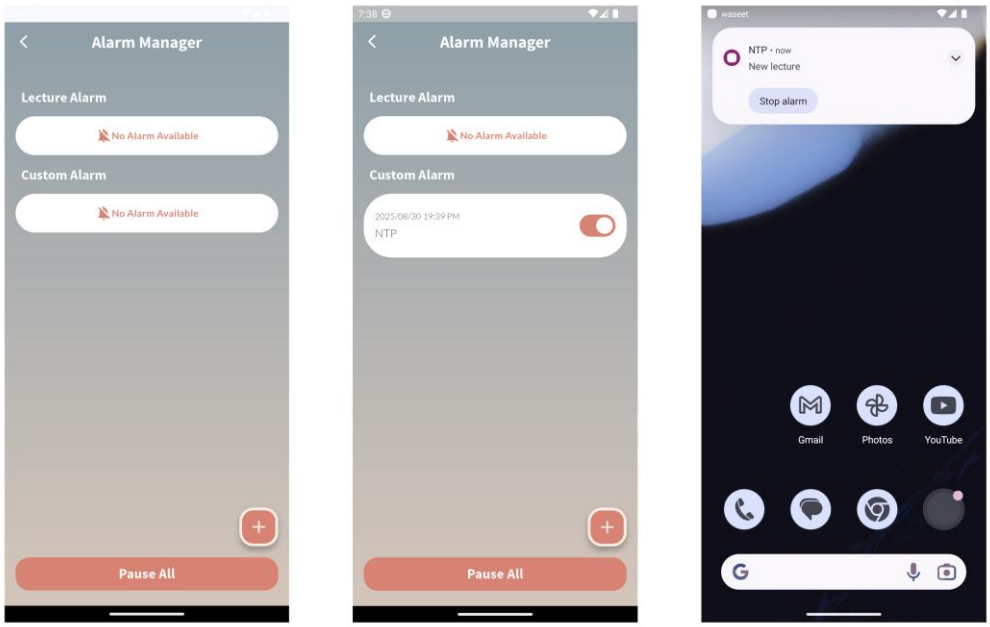
وتظهر فيها معلومات الطالب مثل الصورة الشخصية وعليها مؤشر يحدد نوع الدور وتحت الصورة الشخصية تظهر معلومات إضافية مثل البريد الالكتروني ورمز الدفعة واسم البرنامج ونتائج المواد مع تفاصيلها



الشكل21 – الاختبار – واجهة الملف الشخصي

واجهات التنبيهات:

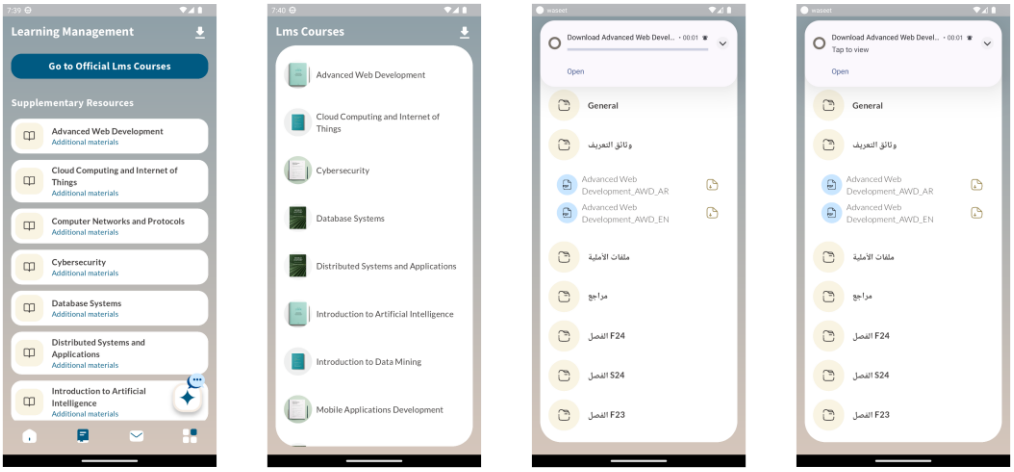
تظهر فيها المنبهات الخاصة بالمحاضرات وتنبيهات المستخدم المخصصة



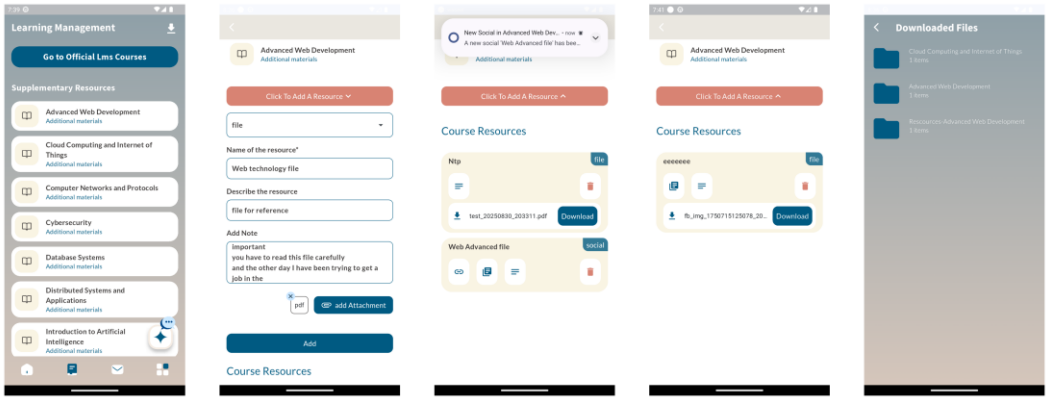
الشكل 22 – الاختبار – واجهة التنبيهات

واجهات إدارة التعلم:

تظهر فيها مصادر التعلم سواء الرسمية أو الإضافية مع خيارات التحميل وروبوت الدردشة



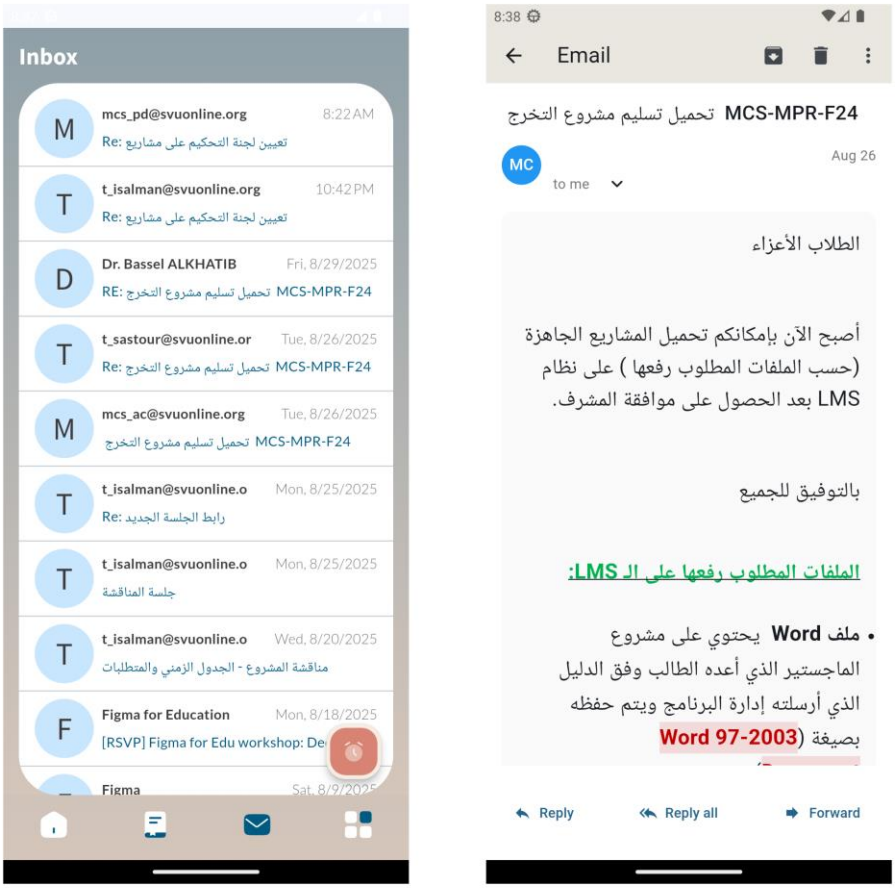
الشكل 23 – الاختبار – واجهات ادارة التعلم 1



الشكل 24 – الاختبار – واجهات ادارة التعلم 2

واجهات البريد الالكتروني:

يظهر فيها صندوق البريد الالكتروني ومنه يمكن عرض أي رسالة كما تظهر الصورة التالية



الشكل 25 – الاختبار – واجهات البريد الالكتروني

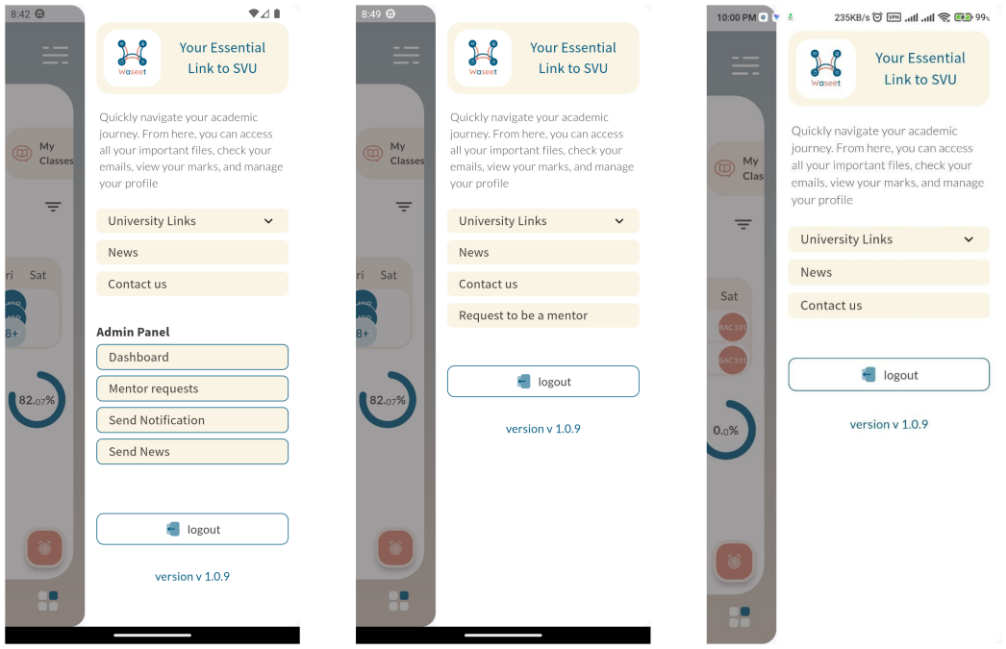
القائمة الجانبية (Drawer):

تظهر فيها مجموعة من الخيارات تتغير بحسب الدور الخاص بالمستخدم (مدير–مرشد–طالب)

والمشترك فيها هو روابط الجامعة المهمة والأخبار وزر تسجيل الخروج بالإضافة الى رقم الإصدار الخاص بالبرنامج

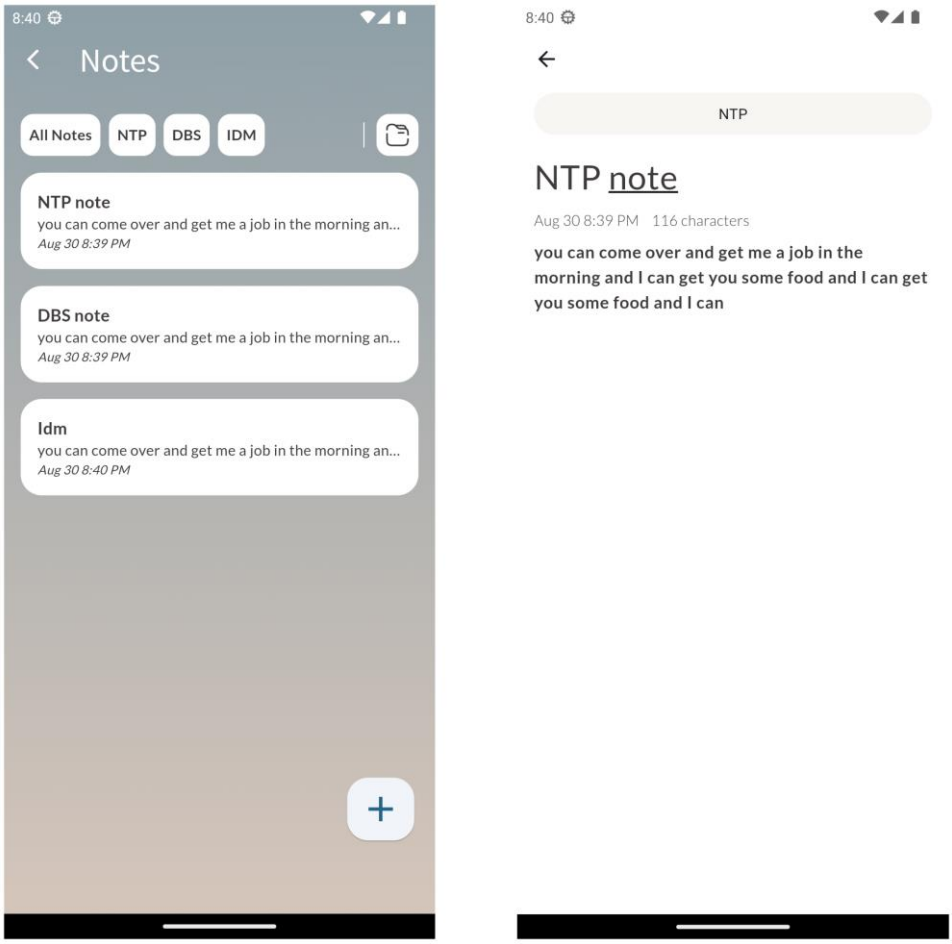
بينما يظهر فقط لدى الطالب خيار طلب الارشاد والذي يختفي في حال أصبح مرشد

بالنسبة للمدير تظهر له لوحة الإدارة وخياراتها كما هو واضح في الصورة هي لوحة التحكم وإدارة طلبات الارشاد ثم خيار لإرسال الاشعار الجماعي لجميع المستخدمين ثم خيار ارسال الاخبار



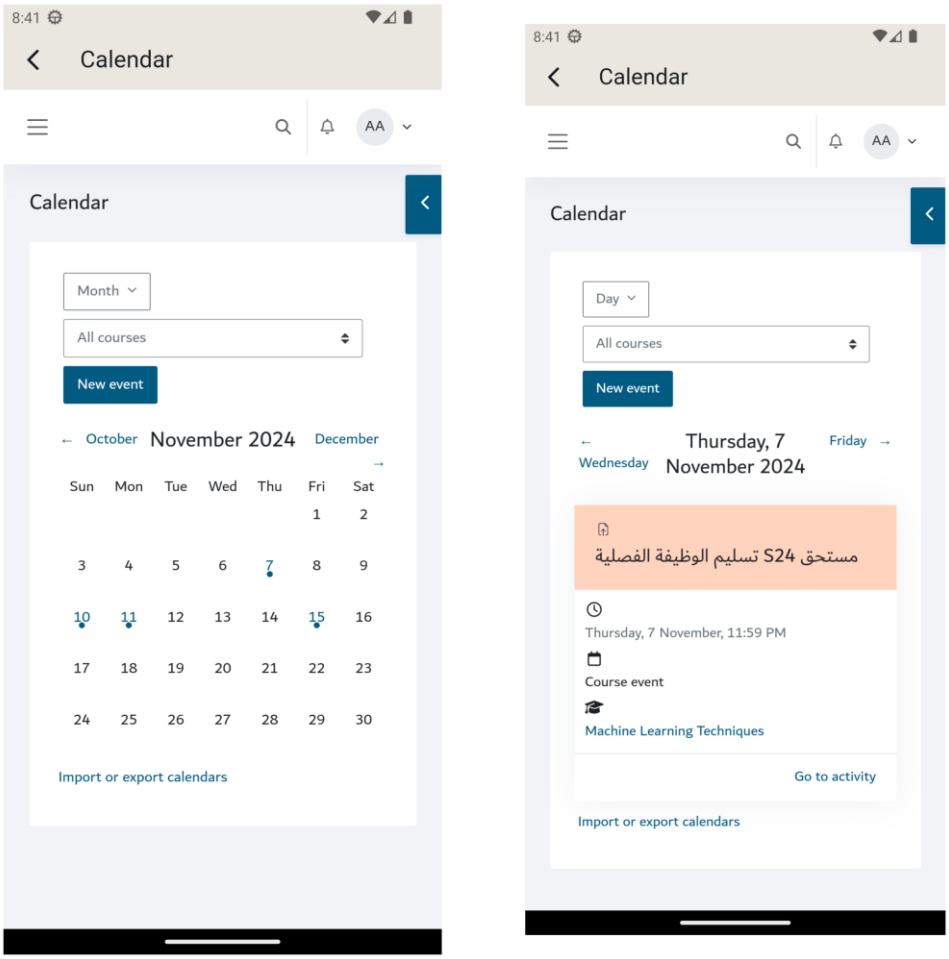
الشكل 26 – الاختبار – واجهات القائمة الجانبية

الواجهات الخاصة بالمفكرة:



الشكل 27 – الاختبار – واجهات المفكرة

واجهات التقويم:



الشكل 28 – الاختبار – واجهات التقويم

الفصل السادس: النتائج والتوصيات

النتائج:

قمنا في هذا المشروع بإنشاء نظام مساعد للطلاب مزود بمميزات متعددة وقابل للتطوير لاحقا واضافة ميزات جديدة، لكن من أهم المميزات التي توصلنا اليها:

- تسهيل الوصول للمعلومات الأساسية للطالب مثل معلومات الملف الشخصي والصفوف وسجل المدفوعات ومواعيد ونتائج الامتحانات بشقيها العملي والنظري.
- عرض الجدول الأسبوعي مع معلومات وتفاصيل كل مخاضرة.
- نظام التنبيهات: وهو من المميزات الهامة حيث يسمح بتفعيل التنبيهات على مواعيد المحاضرات وبالإضافة لذلك يمكن للطالب وضع منبهات مخصصة.
- إدارة التعلم: في هذا القسم من البرنامج قمنا بتوفير المحتوى الرسمي للمقررات من موقع الجامعة وأضفنا اليه قسم خاص بالمحتوى الإضافي الذي سيقوم بتوفيره مجموعة من المرشدين المتطوعين من الطلاب الحاليين أو القدامى لإثراء العملية التعليمية الى حد كبير وتحسين التجربة التعليمية، مع التنويه الى استخدام ميزة الاشعارات للتنبيه عند إضافة محتوى من قبل المرشدين حيث يتم ارسال اشعار فقط للطلاب المشتركين بهذا المقرر للانتباه الى الإضافة والدخول الى قسم التعلم.
- دمج الذكاء الصنعي: من الميزات التي أضفناها الى قسم التعلم دمج روبوت محادثة يؤمن للطالب الإجابة عن الاستفسارات ويعزز من مركزية التعلم ضمن هذا التطبيق فلا حاجة لاستخدام تطبيق مختلف الامر الذي يعزز جودة تجربة المستخدم.
- الملاحظات: ميزة إضافية تعزز من الاعتمادية على التطبيق وتحقق تجميع الميزات في مكان واحد حيث يمكن للطالب إضافة ملاحظات وتجميعها وتصنيفها في مجلدات الامر الذي يغني أيضا عن التنقل بين عدة برامج.
- قسم البريد الالكتروني: وهو أيضا من المميزات الهامة حيث يمكن للطالب من خلال نفس التطبيق متابعة بريده الالكتروني وبفضل ميزة الاشعار بوصول بريد الكتروني لن يفوته قراءة أي بريد هام في الوقت المناسب.
- التقويم: يؤمن الوصول المباشر والسريع للتقويم الخاص بالجامعة ومراجعة المواعيد والاحداث الهامة.
- الارشاد: في المجتمع الطلابي المبادرات التطوعية محببة ونجدها منتشرة بشكل كبير في المراحل الجامعية المتقدمة حيث يبادر احد الطلاب لمساعدة زملائه الحاليين أو الطلاب من الدفعات التالية ويشارك خبراته ونصائحه والموارد التي وجدها مفيدة وهنا قمنا

بتضمين هذه الفكرة في دور المرشد (Mentor) وبالتالي يمكن للطلاب الذين يرغبون بالمشاركة ارسال طلب ليصبحوا مرشدين في برامجهم ويقوم المدير بمراجعة الطلبات والتواصل معهم ثم الاستجابة بالقبول وإعطاء صلاحية المرشد للطالب او الرفض ويمكن أيضا سحب الصلاحية من أي مرشد في أي وقت من واجهة إدارة المرشدين

- الإدارة: يتوفر للمدراء لوحة التحكم الخاصة والتي تتضمن مجموعة من الخيارات أولها لوحة إدارة خدمات الخلفية كما يوجد أيضا ميزة ارسال اشعار جماعي لكل مستخدمي التطبيق للإبلاغ مثلا عن حالات توقف سيرفرات الجامعة او عند وجود تحديث للتطبيق او في أي حالة يجدها مهمة وأيضا ميزة ارسال الاخبار والتي تندرج ضمن عدة تصنيفات مع توجيه اشعار بالخبر حتى نتلافى احتمال عدم الانتباه مثلا في حالات تأجيل او تغيير مواعيد الامتحانات او غير ذلك من الاخبار الهامة وقمنا بوضع مؤشر أولوية يمكن من خلاله تخصيص مثلا ارسال اشعار فقط للأولويات العالية.

التوصيات والآفاق المستقبلية:

على الرغم من تنوع وجودة المميزات والوظائف التي تم تضمينها في هذا النظام لكن بحكم الوقت والموارد المحدودة فلم نتمكن من تضمين كل الأفكار في هذا المشروع بحكم ان بعضها يحتاج الى تجهيزات تقنية أو الى موافقة وتنسيق مباشر مع إدارة الجامعة وقسمها التقني او الى وقت طويل في التطوير لكن يمكن اعتباره هذا التطبيق نواة يمكن البناء عليها والمتابعة بتطويره لاحقا لإضافة مميزات ووظائف جديدة او التعديل على الميزات الموجودة، وبالتالي يمكن القول ان من اهم التوصيات التي يمكن لحظها في عمليات التطوير القادمة:

- في حال نالت فكرة التطبيق استحسان واعجاب الجامعة يمكن الاستفادة من خبرة القسم التقني والتنسيق لرفع المشروع على سيرفرات الجامعة بدلا من الاستضافة المجانية المحدودة وبالتالي تأمين عمل مستقر لخدمة اشعارات البريد الالكتروني وأيضا تأمين واجهات ربط برمجي رسمية لتعزيز إمكانيات التطبيق وتوسيع مميزاته لتشمل جوانب أوسع من المميزات الحالية
- يمكن أيضا تأمين استضافة مدفوعة تحقق قدرات حوسبة اعلى ومساحة تخزين أكبر لتحسين أداء التطبيق وإمكانية استخدام ميزات غير متوفرة في الاستضافة المجانة مثل المهام المجدولة والتشغيل في عمليات منفصلة.

- يقوم التطبيق حاليا وبعد أخذ الموافقة المسبقة من المستخدم بجمع مراسلات روبوت المحادثة الذكي وتخزينها في قاعدة البيانات لنقوم لاحقا باستخدامها في تدريب نموذج ذكاء مخصص للحصول على نتائج أدق حيث حاليا نقوم باستخدام نموذج مجاني من جوجل عبر (api) كخطوة أولى.
- توسع التطبيق لمحاولة تغطية كافة الخدمات الخاصة بالجامعة
- التكامل بشكل كامل مع نظام البريد الالكتروني بما يسمح أيضا بإرسال الرسائل بالإضافة لاستقبال وقراءة البريد.
- إمكانية إضافة ميزة تشغيل المحاضرات التزامنية في وضع عدم الاتصال أو حتى التشغيل المتزامن للمحاضرات في حال تم تأمين الدعم التقني المناسب ودمج نظام تتبع الطلاب من نسبة حضور ومشاركة وعندها يمكن جمع هذه البيانات وتدريب نموذج تعلم آلة للتنبؤ بمؤشرات أداء الطالب او اية معلومات أخرى قد تهم الجامعة.

المراجع

المراجع:

الساطي، طارق, 2022، مقرر أساسيات البرمجة، الجامعة الافتراضية السورية

الشهابي، طلال, 2023، مقرر مبادئ هندسة البرمجيات، الجامعة الافتراضية السورية

المصطفى، محمد مازن, 2024، مقرر تطوير تطبيقات المحمول، الجامعة الافتراضية السورية

الخطيب، باسل, 2022، مقرر برمجة الويب 1، الجامعة الافتراضية السورية

الخطيب، باسل, 2024، مقرر تطوير الويب المتقدم، الجامعة الافتراضية السورية

(1) رابط تطبيق "MU student login" على متجر (Google Play):
https://play.google.com/store/apps/details?id=com.mefgi.login

(2) رابط تطبيق "myUniSannio" على متجر (Google Play):
https://play.google.com/store/apps/details?id=it.cineca.app.unisannio

زينب حسن الشربينى (2012), فعالية تكنولوجيا التعلم النقال لتنمية مهارات أعضاء هيئة التدريس في تصميم المحتوى الإلكتروني ونشره، ماجستير، جامعة المنصورة، كلية التربية, تكنولوجيا التعليم.

Helmy, W. and MA Lashin, M., 2021. مميزات التصميم الجديدة للتطبيقات المحمولة UI/UX للهواتف الذكية. *مجلة العمارة والفنون والعلوم الإنسانية*, *6*(25), pp.480-491.

مجدي محمد ابراهيم, رنا, سلام, سراج and أمل, 2017. أهمية ارتباط تصميم تجربة المستخدم بالتصميم الجرافيكى لإنتاج تطبيقات تفاعلية (مثال تطبيقي على اجهزة الهواتف الذكية). مجلة

بحوث التربية النوعية, 2017(46), pp.689-711.

المواقع الالكترونية:

موقع Classter , "لماذا تعد تجربة الهاتف المحمول الجيدة أمرًا أساسيًا لتحفيز مشاركة الطلاب" تاريخ المقال: 24 سبتمبر 2019 تاريخ الاطّلاع: 2025/08/06.

موقع واكب Wakp, "نظام-ادارة-الطلاب" متاح على: https://wakp.net/ تاريخ الاطّلاع: 2025/08/06.

التوثيقات الرسمية:

- Django Documentation
- Django Rest Documentation
- Flutter Documentation
- Python Documentation
- Dart Documentation
- Python Anywhere Documentation