UNIVERSITY
OF WOLLONGONG
IN DUBAI

SATURDAY'S

FINAL REPORT

Hamzeh – 7392904
Jouman – 7393453
Majd – 7068980
Masooma – 6841545
Mohamed – 7011659

Table of Contents

Introduction

Background information:

As young adults start their academic or professional lives, they often experience problems adapting to 3 important skills: Time Management, Task Management and Communications. Skills that translate to 3 tools in our digital age: Calendar, To-Do Lists & E-Mail. These 3 tools are adapted by every educational institute and every professional Organization. Many fail to grasp these tools and that follows them into their professional lives and even after adapting to said tools, conflicting tasks & events, and the overwhelming number of emails one may find in their inbox after a long weekend, or a vacation can prove impossible to navigate with university professors being a prime example of that.

Problem statement

Time, Task, and Email management are skills that can be overwhelmed by the sheer number of responsibilities, commitments and emails one might find themselves having to deal with. . The typical solution is a personal assistant, a skilled professional which dedicates their effort to managing and maintaining. As amazing as that sounds, personal assistant costs money… a lot of money for a typical academic, young professional or even a university professor. An alternative would be a “Virtual Personal Assistant” which does the same job of a personal assistant for less money for working remotely, however that is still a $200 – $400 bill which not only can be infeasible, but often proves to be an expensive experiment until a good match has been found due to the language and time zone differences.

Our solution

An AI assistant has numerous, significant advantages. First of all, it provides an invaluable way to get through the complex maze of obligations that now characterize our everyday lives. This technology partner excels at streamlining intricate procedures and releasing users from the constraints of laborious, manual labor by utilizing sophisticated algorithms and data-driven insights. According to “Computers in Human Behavior” (Huang and Rust 2020), the outcome is a more efficient and well-structured daily schedule that maximizes time management and reduces the possibility of important tasks being overlooked. Second, the AI assistant is made to accommodate each user's distinct and varied needs. It is not a one-size-fits-all solution, but rather a flexible and user-friendly tool thanks to its configurable and customizable features. It understands and meets individual needs, whether they have to do with effectively managing calendars and events or methodically sorting important emails into manageable to-do lists.

In the end, an AI assistant is a faithful friend in a world full of complexity and demands. The quality of life in a world that is becoming more complex and interconnected is improved as a result of it easing the burden of daily life, enabling people to reclaim control over their time, and improving overall productivity and organization.

Through the following features, we hope to further personalize our AI system so that it can easily adjust to each person's unique daily routines and obstacles, increasing productivity and relieving the burden of hectic schedules. The features we will be working on are:

1. Calendar Management: This AI assistant's efficient calendar management is one of its main features. Through the organization and prioritization of crucial appointments and commitments, it helps users maximize their schedules and lowers the likelihood that crucial tasks will be missed.

2. Email to-do lists: With so many messages arriving in inboxes, it can be difficult to sort through them all and find the important stuff. By generating to-do lists and extracting pertinent contexts, the AI assistant assists users in managing their emails and makes sure that critical tasks are completed on time.

This AI assistant is a flexible productivity tool designed to help users take back control of their everyday life, reduce stress, and maximize their time and

resources. It gives people an easy way to combine all of their daily responsibilities and tasks, which improves their productivity and organization.

Target Niche

Students and university professors are a gold mine of data for the future development of this project. As they are abundant in the environment of the project, they are prime examples of lack of experience of managing such responsibilities when it comes to first year students, and overwhelmed professors and senior students.

Said demographic sets itself as the plausible ideal consumer, as they are unable to employ a full time staff member dedicated to managing their calendars and emails, yet in sever need of such a service.

Technologies

For Saturday to take shape there are some aspects that are in need to be addressed.

- **Application Design**: Defining the method of user interaction and its overall design.
- ***Machine Learning**: Identifying and researching the algorithms and techniques that would allow Saturday to gain a better understanding of its user's priorities and learning to define important emails.
- ***Datasets:** Finding the right data sets that would allows for a smoother user to AI communications, a dataset that would allow the AI to learn calendar management from event generation to event management. NOTE: Some difficulty is expected when searching for relevant datasets and it might end up with the need of building said dataset(s).
- **NLP:** A deep understanding of the ins and outs of the NLP models is imperative for the team to gain the ability to develop Saturdays very own models.

- **Algorithm development:** Developing the algorithm which will allow Saturday to compute the best possible scheduling results.
- **Stable Internet Connection:** Saturday is designed to be a cloud based service which means a stable and secure internet connection will be needed for users to access it's services.
- **Software, Hardware & Languages:** Identifying the software and hardware requirements is a crucial step as it allows to compare and contrast the different service providers and allow higher clarity of choice. Similarly understanding the different libraries and capabilities of different programming languages is imperative to identify which of the languages is best suited.

Feasability

Project charter

1. Project Information		2. Business Case
Project Name:	Saturday	Many people fail to manage their tasks and calendars effectively, resulting in missed deadlines, scheduling conflicts, and general productivity issues. Existing task and calendar management apps frequently lack intelligent features and automation capabilities, requiring users to manually enter and organise data.
Project Description:	AI-powered assistant that helps with tasks like task prioritisation and management.	

3. Project Deliverables	4. Project Benefits
Feasibility Report	Enhanced productivity and time management with intelligent work prioritisation and schedule optimisation.
Requirements Document	Enhanced organisation and visibility into tasks and deadlines, reducing the likelihood of missed appointments or commitments.
Design Document	Automating repetitive chores like email processing saves time and reduces cognitive stress.
Testing Document	Improved consumer satisfaction and engagement with personalised recommendations and an intuitive user interface.
Integrating the components	
Deployment of the application	

5. Project Risks

Risk: Competition from similar existing apps in the market
Mitigation:
Market Differentiation: Identifying unique selling points and features that set the app apart from competitors.

Focusing on adding value or addressing specific issues that present apps do not effectively address.
Marketing and branding: Developing a strong marketing strategy to raise awareness and visibility of the app among target customers.

Using digital marketing platforms, social media, and partnerships to reach a larger audience and setting the app apart in a congested market.

Risk: Integration issues while connecting the different components of the application
Mitigation:
Creating standardised data formats and APIs for transferring information across components.
Conducting extensive testing on data transmission and processing to detect and resolve any integration difficulties early in the development phase.

Risk: Inaccuracies or errors with the AI powering the chatbot and email processing functionalities
Mitigation:
Conducting extensive testing and validation of AI using a variety of scenarios and inputs.
Implementing measures to continuously analyse and enhance AI performance based on user feedback.

Risk: Low user adoption or engagement
Mitigation:
Conducting user research and usability testing to ensure that the app's interface and functionality fit the needs and preferences.

Risk: Failing to comply with relevant data privacy regulations
Mitigation:
Conducting a thorough examination of applicable data privacy rules to ensure that the app's data handling processes meet legal requirements.
Implementing features like data access controls and consent processes to provide users more control over their personal information.
Conducting regular audits of the app's compliance status and adjusting policies as needed to reflect regulatory changes.

Risk: Developing an accurate and efficient AI model
Mitigation:
Ensuring that the training data is high-quality, relevant, and diverse.
Performing extensive data validation and verification to detect and resolve any issues that may affect model performance.
Using regularisation techniques to reduce overfitting and increase model generalisation.
Researching ensemble learning techniques, to increase the model's prediction accuracy and robustness.

6. Project Budget	
Provide an estimated budget for the project, including breakdowns for specific tasks or resources	
Total Budget: $ __1,000,000 ~ 2,500,000 __	
Breakdown: AI development, Data related (Storage, handling, etc..), Infrastructure Costs, Other Costs (Research, testing, UI/UX design)	
AI development	$ 120,000~500,000
Data	$ 300,000 ~ 600,000
Infrastructure	$ 100,000 ~ 250,000
other	$ 80,000 ~ 250,000

7. Project Milestones	
Project Initiation	03.21.24
Market research, Requirements research and gathering	04.21.24
Design Phase	05.21.24
AI Model Development	08.21.24
Front-End Development	9.21.24
Back-End Development	11.21.24
Testing phase	1.21.24

Deployment Phase	2.21.24
Market Launch	3.21.24
Post-Launch	4.01.24

8. Project Stakeholders	
Role	**Expectations**
Faculty/Professors	**Expectations:** Guidance and supervision throughout the project to ensure academic rigour, adherence to project rules, and demonstration of learning outcomes. **Requirements**: Provide regular progress updates, adhere to the project timetable, complete all required documentation (e.g., project proposal, progress reports, final report), and deliver project outcomes.
Project team members	**Expectations:** Successfully completing the project, demonstrating technical skills and knowledge, and meeting learning objectives. **Requirements**: Collaborative working, adhering to the project strategy and deadline, maintaining regular communication, completing given tasks and deliverables, and participating in project presentations and demonstrations.
End users	**Expectations:** User-friendly interface, effective task and calendar management capabilities, and consistent performance. **Requirements**: Intuitive app design, sensitive to user feedback, customisable settings, and privacy and security measures to secure user data.

Future Investors	**Expectations:** Return on investment, possibilities for future funding or partnerships, and meeting project deadlines and objectives. **Requirements**: A clear project plan and budget, regular reports on project progress, and adherence to reporting criteria and deliverables specified in financing agreements.

Scope statement

The aim of this project is to create an application that would act as an entire system for its user. Managing one's emails, calendar and tasks is one of the biggest challenges for young adults in their higher education journey. Even educators with all their experience may be overwhelmed with the ever so increasing number of students they have every semester let alone year! Hence “Saturday” an AI dedicated to aiding you in accomplishing your tasks and scheduling them for you! Utilizing an AI module trained to assess the content of emails received, “Saturday” will:

1) Assess possible tasks, events or deadlines the user is being informed of. a. Tasks: break down the task and schedule working hours. b. Events: Allocate the time slot specified for the event on the user's calendar. c. Deadline: Allocate reminders and calendar time slot allocation on the user’s calendar.

2) Allocate importance and urgency levels to emails in the inbox. (Raising awareness to important emails.) Generating an “Actionable Email” list.

3) Generating list of tasks ensuring the completion of an “Actionable Email” within the To-do list.

4) Allocate time slots from the user's free time in accordance with their calendar.

Goals & Objectives

1) Creating a fun, intuitive and innovative front end to attract users & ease the user adaptation process.
2) Establishing a secure and robust backend system allowing the retrieval of user emails and connection to Saturday’s AI, Front end and the Service providers servers.

3) Establish a well-trained AI model.

PESTEL Analysis

A Break down analysis of the Political, Economical, Sociological, Technological, Environmental & Legal aspects of “Saturday’s” current & future development in the UAE.

Politcal:

AI Government Policies: As of the current day, the UAE is aiming towards accelerating the adaption and development of AI in the UAE. It has taken many steps in said effort including stratigic partnerships in both the private & public sectors.

National AI Councils: The establishment of multiple Artificial Inteligence focused councels which is tasked with the oversight of the integrationa and adaptation processes of AI in both the government & the education sectors.

AI Ethics: Under the guidance of the UAE government, currently there exists rules, priniples, & guidelines any system integrating AI must adhere to in the UAE.

AI Ethics: To ensure a smooth integration into the UAE market and a proper development phase in the region. “Saturday” must adhere to the AI Ethics Principles and Guidelines propesed by the UAE government.

Economical:

Market Demand: Productivity tools are an every lasting market, with its along with its technological aspect and the scalability of software solutions. This product has both a large market to attain and an even larger pool of products to compete with. While AI is slowly growing and expanding its hold over the market, a student specific AI tools is yet an untapped market with large potential.

Economic Conditions: The economy of the launching region/country will be an integral part of the products lunch and monotizability. However with the large number of univeristies based in UAE with backing of the government, the various international univeristies with satellite campuses in the UAE. Both the economical and network conditions are in their prime conditions for such a product to launch.

Digital Econonmy: "The strategy aims to double the contribution of the digital economy to the UAE’s non-oil GDP from 11.7 per cent to over 20 per cent within the next 10 years." UAE Cabinet, Digital Economy Strategy.

Sociocultural:

Digital Literacy: “Saturday” must ensure to introduce an intuitive user interface to attract & maintain users and ease the apotion process by the user base. With the target audience being students in the age range of (15-20) the

likelihood of the digital literacy being in between 95-100% ensuring a digitally literate target costumer avatar.

Work-Life Balance: With the "Start-up work culture" found & expected in our post-Covid world, the UAE & primarily in Dubai a strive for a healthier work life balance is at its highest levels. Hence the need for an application that would adiquitly allocate the time and priority of tasks is highly saught after.

AI Education: The UAE's support of the integration of AI is evident by the training programms the government has offered its officials, private sector employees and the residence of its country in AI specialization courses. Which not only displays the welcome for such a product into its market but the ever increasing awareness and familiarity of AI in the coustomer base.

Technological:

AI Advancements: "Saturday's" evolution will be accelerated & accompanied by the development of the global AI technology, which as shown an explosive growth in the industry has seen in the past 3 years.

Cyber-Security: As the application will have to deal with sensetive personal information of the user (the users emails) the robustness of "Saturday's" security aspect has to be established and well maintained thorughout the application lifespan to ensure capture the userbase and costumer base's trust.

Data Storage & Accessability: Various data storage and accessability laws & regulatoins must be adhered to in the further stages of "Saturday's" development to ensure ease of transition into the market, perticularly being the UAE's Laws and regulations.

Evironmental:

E-Waste: Ensuring a minimal level of enviromental impact has been made easier with the digital nature of "Saturday".

Energy Consumption: Optimization of the energy consumed by the servers used by "Saturday" shall be an integral aspect of the applications future indevours.

Sustainability:

"Satuday" must allign itself with the UAE's commitment to sustainability.

Legal:

AI-Based Laws: "The UAE is developing a platform for designing financial laws using digital solutions, known as the "Rules as Code" platform." (UAE Government)

Data Protection Laws: "The UAE has implemented the Personal Data Protection Law, Federal Decree Law No. 45 of 2021, which your application must comply with." (UAE Government)

Consumer Protection Law: "The Federal Law No. 15 of 2020 on Consumer Protection protects all consumer rights, including the data of the consumers and prohibits suppliers from using it for marketing." (UAE Government)

The references for the PESTEL Analysis can be found under the "References" section at the end of the document.

Competitor solutions and their limitations

1. Calendly: Allows users to share a link with others to book time blocks. However, it does not automate time management. Event generation is not automated based on emails, but delegated to the other parties allowing events from emails and Calendly events to clash. And it does not aid in task management (specifically generating ToDo lists).
2. Clockwise: A time management tool which creates time blocks on your calendar for focus sessions and notify team members of user's availability. Its limitations include; lack of task management tools, lack of calendar management, and inability to transform emails into calendar events.
3. Google Bard: Google bard offers an AI chat bot with some access to the google drive and Gmail services. It would be theoretically able to generate a to do list based the email specified in the prompt. However not only is this not a built-in functionality, it currently has no access to the google calendar to create a task entry and there is no time nor task management functionality built in.

Work Breakdown Structure

Background info about topic (Juman), Project Description and Objectives (Juman and Hamzeh), Project Scope (Mohammed Shar and Majd), How we do it (Juman), Software and Hardware Requirements (Mohammed Shar and Majd), Challenges (Majd), Future Improvements (Masooma), Timeline (Hamzeh and Mohammed Shar), Project Management Tools to be used (Juman and Masooma).

Milestones and division of work among team members

Back End

Week 3: The back end functionalities of the project along with the APIs and the documentation was done. Further testing and optimization was needed but prototype level of completion was completed.

Week 4: AWS server setup was complete and start of migration of environment to AWS servesrs started.

Week 6: Documentation of the functions, error handeling and security optimization was emplemented and completed.

Week 9: OAuth (A security & 3 party “Google” implementation) was established allowing secure access to user Gmail account.

Week 10: Further enhancments and function optimization & testing.

AI

Week 1: exploring types of models, types of classifiers and looking for datasets.
Initial model: a combination of SVM and Random Forest. Understanding the concept of text classsification, and feature extraction

Week 2: finding a text classifier, understanding how the classifier works. Understanding the concepts of LLM

Week 3: looking for other models, looking into IBM Watson chatbot

Week 4: looked into transfer learning, searched for datasets. Working on the email classification algorithm

Week 5: Data set and AI model was selected and traning process has started.

Week 6: Preprocessing the dataset to handle the outliers.

Week 7: Testing the model and improving the accurarcy

Week 9: optimizing the final model and enhancing its features

Week 10: final testing and integrating the components together.

Front End

Week 1: Designing the initial front-end, choosing the color palette and design.

Week 2: finalising the final design of the front-end.

Week 3: Creating the login and signup pages.

Week 4: Creating the calendar and to-do list pages.

Week 6: Fixing the errors with the front-end.

Week 8: Testing the front-end.

Week 10: Integrating the front-end with other components of the application.

SWOT Analysis

STRENGTHS +	WEAKNESSES –
Innovative AI-powered features Technical Expertise	Resource constraint Technical dependencies Team coordination Limited experience with AI development

OPPORTUNITIES +	THREATS –
Demand for productivity tools . Potential partnerships . Expansion into new markets . Integration with other platforms	. Competition from existing apps . Technological Advancements . Regulatory changes . External uncertainties

Risk Management Plan

Identifying the risks:

Working with AI or perosnal data can open up multiple risk factors. Especially when pairing these 2 technologies and serving them up to such a young target audience which dont often follow the best secuirty practicies. The risk factors that would be imposed on projects such as “Saturday” would be:

- **Data Base Security:** If the databaes holding the users private emails is compromised without a proper mitigation plan, action plan, & execution “Saturday” would most probably lose the trust of the stakeholders (Consumer base, investors and business partners form univerise & other invested parties)
- **AI Server Compromise:** A compromise to the server witholding the legacy AI could lead to a leak of the legacy code, making the monotization of “Saturday” practically impossible creating a loss to the stake holders in a financial aspect and trust in the porject and its developers.
- **AI Safety Compromise:** The saftey of the user when interacting with “Saturday’s” legacy AI is imperative, as the user experiences, prodcut effectivness and overall usability of the application are all relying on the

AI to work according to its design and for it to fail to meet stakeholders expectations would be failure to meet the market's standards, needs and expectations.

- **System Compromise:** "Saturday's" development must ensure a robust end product that can withstand various user inputs from malicous inputs such as; SQL injections, malicous prompts, and other inputs which aim to cause harm to any aspect of the porject and its user.
- **User Data Compromise:** The security of the user data must be kept along the priorites of the projects security aspects as the end user can either uplift or create an end ot the project's lifespan due to their experience. The physical and digital security of the user data must be maintained and ensured.

Managing The Risks:

Data Base Security Mitigation Plan: To ensure the safety of the users database, "Saturday" may outsource its data storage to a reliable & reputable third part which would allow the transfere the risk to a more experienced and prepared party. Upon further development and research the storage of the user data could move to a local database maintained by the in-house team. Alternatively the data could be stored on the user machine ("Smart Phone") which would need further research and development to ensure the saftey & feasability of both options. Further research could also reveal other solutions.

AI Server Compromise Mitigation Plan: dsa To ensure the security and reliability of "Saturday's" AI hosting servers, out sourcing the service would allow a transference of the risk, while allowing to both increase the timeframe to move ot a locally maintained and ran server while elapsing the products lunch timeframe. Both of which will allow for more data & more time to assess the optimial runtime security, processing power, electric consumption and the

various other aspects of AI hosting server. This will allow for minimal spediture, electric and runtime waste in further phases.

AI Safety Compromise Mitigation Plan: Various methodologies of testing will be required to assess the numrous risk factors imposed on such an application from Acceptance testing, pen-testing, prompt engineering & reverse engineering testings. Based on the tests findings a proper mitigation plan will be further developed. This testing phase could be outsourced to a renown third party vendor which will be more experienced and through than the original development team due to experience and variety of sepcialists. Upon further testing the results will show how the code will react to various unecxpected inputs, the AI will reveal any and all prompt engineering threats, and sercuity of the data will be assessed propery and based on that the serurity of the system and the user data will be accurately assessed and proper plans of action could be drawn.

AI Safety Compromise Mitigation Plan: Poper & thorough tesiting of the AI model, its engineering, training and its responses would be in need in further development phases especially before going to market with the product, ensuring the a safe to use AI model which will stand strong in the face of wreckless & malisous users. Ensuring the accuracy and dependancy of the AI and it's responses.

System Compromise: Proper and thorough testing of the system in all of its aspects is highly needed for the further development of "Saturday" as it must be able to withstand both malisous and wreckless handeling from its users and respond accordingly. Due to the nature of the intended coustumer base's unpredictablity both the AI & the back end must be propery tested and secured from; Malisous prompts & wreckless access and usage of the product. Outsourcing the job of testing will be highly effective as it ensures an unbiest examination and assessment of the product by an experienced party ensuring the product's integrity remains intact when faced by such a user base.

User Data Compromise Mitigation plan: The user's trust in the system is directly correlated with the saftey of the users data, therefore whichever approach taken to collect, store and access the users data whatever it might be must not only adhere to the laws and regulations of the country and region of the products launch and development loation, but also ensured to capture and maintain the stakeholders trust and satisfaction. Thorough testing of the chosen approch is needed from pen-testing of the Data base to ensure the CIA triad of the data while in storage and in transit.

Regular checkups, audits, updates, upgrades & software patches are definetly expected to ensure the products maintenance and preservation of its lifespan.

Requirement analysis

Definitions and assumptions (M.shar definitions)

Definitions:

Assumptions:

Saturday's functionalities will depend on machine learning algorithms and datasets that will be made once proper development begins. Saturday will protect user data by following the data privacy regulations and guidelines.

The development of the User Interface (UI) and User Experience (UX) depends on the design and layout of the application. Choices made will consider better user interaction and overall user satisfaction.

The assumptions and dependencies mentioned above are to be reviewed, validated and updated as the project progresses to ensure accurate and successful execution.

Design constraints

The following constraints are few that could impact Saturday's functionalities. However, this can change based on the development and implementation.

1. Platform Compatibly: Saturday will be compatible with both IOS and Android Devices

2.Language Support: Saturday does not support multiple languages and cannot cater to a more global user base.

3. Third-Party Integrations: Saturday needs to integrate third-party services to fully achieve its functionality.

Example: Gmail does not allow third-party services, making it incompatible as an email client.

Functional requirements

Login

Name

Login

Goal

This feature allows the user to log in to their account enabling access to the applicaϴon's services and access to their data.

Input

1. Username

2. Password

3. Two- factor authentication code (optional)

Output

The application will either

1. Show the user a successful message indicating that the user has been successfully verified, and then the user be redirected to the home page.

2. Show the user an error message indicating that one or more of the requirements inputs are wrong.

Main scenario

A user would need to log into their account to access "Saturday's" services. Ensuring fluency and consistency of data across different host machines.

Pre-conditions

1. The user must have the application installed.

2. The user must be connected to the internet.

3. The user must have a registered account in the database.

Steps

1. The user will open the application.

2. The user will click on the login option.

3. The application will display two input fields (username, and password).

4. The user will input the username and password.

5. If the user two-factor authentication is on, then a message will be sent to the user preferred channel, and the user will then input the code.

Post- conditions

The user will be granted access to the application, and the user will then be redirected to the home page.

Exceptional scenario

If the user credentials were not verified by the system, or the two-factor authentication is wrong, the system will display an error message to the user indicating that one or more inputs were wrong, the system will then ask the user to reinput the credentials.

Validation <<include>>

Name

Validation

Goal

To ensure the confidentiality of the user's data, they will need to present their accounts credentials.

Input

1. Username

2. Password

3. Two-factor authentication (If opted by the user)

Output

The function will either

1. Notify the application that the credentials exist and are valid in the system database.

2. Notify the application that the credentials either do not exist in the system database, or the inputs are invalid.

Main scenario

When the user submits his/her credentials in purpose to login into the system, the validation

function will check if the credentials exist and are valid in the database of the application, this function will work after the user submits the credentials, and before the application grants or deny access to the user.

Pre-conditions

1. The application and the validaϴon function are online.
2. The user entered his/her credentials in the input fields.
3. The user attempted to login after entering the credentials.

Steps

1. The user input his/her credential into the application login input fields.
2. The user attempts to login to the application.
3. The validation function will check if the credentials exist and are valid in the database.
4. If the credentials are valid and user two-factor authentication is enabled, the validation function will verify the two-factor authentication code.
5. The function will notify the application whether the credentials are valid or not, so the login function can take the necessary actions.

Post- conditions

The validation function will notify the login function that the credentials exist and are valid.

Exceptional scenario

If the credentials were not verified, the function will notify the login function that there is an error, so the login can take the necessary action.

Two-factor authentication <<extend>>

Name

Two factor authentication

Goal

The purpose of this function is to add an extra layer of security for the login function, The function will require the user to input a random generated code after validating the credentials submitted, ensuring that there is no unauthorized access to any user account.

Input

Two-factor authentication code

Output

The system will either display

1. Successfully message indicating that the two-factor authentication code submitted by the user is correct.

2. Error message indicating that the two-factor authentication code submitted by the user is incorrect.

Main scenario

After the user has entered the credentials, and the validation function has validated that the credentials exist and correct, the user will be requiring to enter a random generated code that will be sent to the user preferred channel to complete the login step.

Pre-conditions

1. The user credentials have been submitted.

2. The credentials have been validated by the validation function.

3. The user has access to the channel where the code will be sent.

Steps

1. After validating the user credentials, the system will generate a random code for the two-factor authentication.

2. The user will receive the code through the preferred channel.

3. The user will enter the code in the system to verify the identity.

4. The function will verify the code that the user entered with the code that was generated.

5. The system will grant access to the user to explore the features of the application.

Post- conditions

The user login step will be completed, and the system will redirect the user to the hope page.

Exceptional scenario

If the two-factor authentication step failed, the system will:

1. Display an error message indicating the reason if the error.

2. The user will be prevented from accessing the features of the application.

3. The function will allow the user to resend the code to verify the identity.

4. The function will send an email to the registered email address stating the atempt.

Logout

Name

Logout

Goal

The purpose for this function is to terminate the user active session, ensuring that the user account is not left active for any unauthorized access or other people access.

Input

User clicking on logout button.

Output

The user will be redirected to the login page.

Main scenario

After the user finishes exploring and accessing the features of the application, the user will logout ensuring the safety of his/her account, and the system will clear the user active session.

Pre-conditions

The user must be connected to the system with a valid and active session.

Steps

1. The user will navigate to the place where the logout button is located.

2. The user clicks on the logout button.

3. The system will clear the user active session.

4. The user will be redirected to the login page.

Post- conditions

The user active session will be cleared, and the user will no longer be logged in.

Exceptional scenario

N/A

Reset password

Name

Change password.

Goal

The purpose for this function is to provide the user with a secure method to reset the password in case the user cannot remember the password.

Input

 User ID (Username, Email address, Phone number)

Output

The system will:

1. Send a link to the registered email address or registered mobile number to enable the user to reset the password.

2. Display the instructions on how to change the password on the application interface.

Main scenario

A user who has forgotten the password and is unable to login to the application to access the features will use the function to change the password.

Pre-conditions

1. The user must already have an account that is registered in the system.

2. The user must have access to either the registered email address or registered phone number in the system.

Steps

1. The user clicks on reset password option in the login page.

2. The user will input either the username, email address, or phone number that he/she used to register the account.

3. The password reset ink will be sent to the user preferred channel, with instructions on how to change the password.

4. The user will follow the instruction sent and change the password.

Post- conditions

The user password will be changed, and the user can log in to the applicaϴon with the new set.

Exceptional scenario

If the user account does not exist, the system will:

1. Display an error message indicating the account does not exist.

2. Give the user the option to either change the way of locating the account or contact customer support

Verify user <<include>>

Name

Verify user.

Goal

The purpose for this function is to verify the identity of the user that is attempting to change the password. This function adds an extra layer of security ensuring that only the legitimate user can reset the password.

Input

Password reset code sent to the email address registered or phone number.

Output

The system will:

1. Successfully message indicating that the function has validated the identity of the user.

2. Allow the user to securely reset the password.

3. error message indicating that the function could not validate the identity of the user.

Main scenario

A user that has requested to reset a forgotten password will have to go through the step of verifying and validating the identity by entering the code that will be sent to the registered email address or phone number in the system.

Pre-conditions

User must have already requested to reset the password.

Steps

1. The user must enter the code that will be sent to the preferred channel.

2. The function will validate the code that has been submitted by the user.

3. The user will be granted access to reset the password.

Post- conditions

The user will be granted the chance to reset the password, and the application will display an input field requesting the new password to be set by the user, giving the user access to his/her account again.

Exceptional scenario

If the validation failed due to wrong code entered by the user, the system will:

1. Display an error message explaining the reason for the failure.

2. Prevent the user from resetting the password.

3. Give the user the chance to revalidate the identity.

View setting

Name

View setting

Goal

The purpose for this function is to allow the user to see the current configuration and preferences set in the application.

Input

User clicks on the setting button.

Output

1. Display the current setting set in the application including user setting (privacy, security, account preferences), and Saturday`s setting (color scheme, notification, pause length, priority level, and important email address).

2. Allow the user to edit and modify a specific setting.

Main scenario

A user will access the setting page to review and/or modify how the account is set up.

Pre-conditions

The user must have an account that is registered in the system.

Steps

1. The user clicks on the setting button.

2. The application will request the current setting from the database.

3. The user browses and views the current setting set for his account.

4. The user can modify and change a specific setting to his preference.

Post- conditions

The user views the current setting applied in the application for his account.

Exceptional scenario

N/A

edit Saturday's settings <<extend>>

Name

Edit Saturday`s settings.

Goal

The purpose for this function is to allow the user to customize the current application behavior which includes, color schemes, notification preferences, pause lengths, priority levels, and important email addresses specific to the virtual assistant Saturday.

Input

Users click on the edit button.

Output

Confirmation message indicating that the previous setting stage has been changed.

Main scenario

The user navigates to the setting page intending to change the current stage setting for Saturday, to enhance the user experience and to change the application behavior to the user experience preferences.

Pre-conditions

1. The user must have an account registered in the system.

2. The user must have accessed the view setting page.

Steps

1. The user must be logged in.

2. The user accesses the setting page.

3. The user selects the setting he/she wants to modify and input the new status of the setting.

4. the system updates the setting value in the database.

5. The application displays confirmation messages to the user.

Post- conditions

The modifications that have been made by the user will be saved in the database by the system, and the behavior of the application will change according to the new status of the setting.

Exceptional scenario

N/A

edit user settings <<extend>>

Name

Edit User settings.

Goal

This function allows the user to customize the current personal account settings, including privacy, security, password, email, and account preferences.

Input

Users click on the edit button.

Output

Confirmation message indicating that the previous setting stage has been changed.

Main scenario

The user navigates to the setting page intending to change the current stage setting for user account.

Pre-conditions

1. The user must have an account registered in the system.

2. The user must have accessed the view setting page.

Steps

1. Users access the setting page.

2. The user selects the setting he/she wants to modify and input the new status of the setting.

3. the system updates the setting value in the database.

4. The application displays confirmation messages to the user.

Post- conditions

The modifications that have been made by the user will be saved in the database by the system, and the new personal settings will be applied to the user account and experience.

Exceptional scenario

N/A

view calendar

Name

View calendar.

Goal

This function allows users to view their scheduled events, appointments, and tasks and give reminders and suggestions on how to manage the events in the user's calendar.

Input

User clicks on the calendar open.

Output

1. A calendar with the date and time

2. Events, and appointments that the user has.

3. Reminders on upcoming events and appointments.

4. Suggestions on how to manage the events, appointments, and tasks.

Main scenario

A user wants to view their upcoming events, appointments, tasks, and to get suggesϴons from the virtual personal assistant on how to manage the calendar.

Pre-conditions

1. The user must be logged in to the system.

Steps

1. The user selects the calendar option from the main page.

2. The system gets the user calendar from the database.

3. The application displays the user calendar.

4. The application suggests to the user how to manage the upcoming tasks, events, and appointments.

Post- conditions

The user can view their upcoming tasks, events, and appointments, and get suggesϴons from the virtual personal assistant on how to manage the calendar.

Exceptional scenario

N/A

add event <<extend>>

Name

Add event.

Goal

The purpose for this function is to allow users to add new events, appointments, tasks to their calendar, and get suggestions on how to manage them effectively.

Input

1.Event name

2.Event description

3.Event priority level

4.event timing

5.Event duration.

6. Events notes.

7.Event location

Output

The system will:

1. Display a message indicating that the event has been added to the calendar.

2. Display suggestions on how to integrate the new event to the calendar.

3. The user can view the new event in the calendar.

Main scenario

A user wishes to add an event, task, or appointment to the calendar, and seeks assistance from the virtual personal assistant on how to manage the new event effectively.

Pre-conditions

1. The user must be logged in to the system.

2. The user must be on the calendar page.

Steps

1. The user clicks on the option to add a new event, task, or appointment to the calendar.

2. The user inputs the details of the new event and submits to the system.

3. Saturday will save the new event into the database.

4. Saturday will analyze the new event based on the user calendar and situation and provide suggestions to the user.

Post- conditions

The user can view the updated calendar, with recommendations on how to manage it effectively.

Exceptional scenario

If the user inputs an invalid information or missing information, then the system will:

1. display a message indicating the error to the user.

2. Prevent submission of the new even

delete event <<extend>>

Name

Delete event.

Goal

The purpose for this function is to delete scheduled events, appointments, or tasks from their calendar, and offer a strategy on how to manage the calendar.

Input

1. The user selects the event.
2. The user clicks on delete the event.

Output

Saturday will:

1. Show a message indicating that the event has been deleted.
2. The new updated calendar will no longer have the deleted event.
3. Show suggestions on how to manage the remaining tasks.

Main scenario

A user decides to delete an event, appointment, or task from their calendar and is seeking advice from Saturday on how to manage the calendar after deleting such event.

Pre-conditions

1. The user must be logged in to the system.
2. The user must be on the calendar page.

Steps

1. The user clicks on the option to delete an event, task, or appointment from the calendar.
2. Saturday will update the new user calendar into the database.
3. Saturday will analyze the new calendar after deleting the event and provide suggestions to the user on how to manage the calendar accordingly.

Post- conditions

The user can view the updated calendar, with recommendations on how to manage it effectively.

Exceptional scenario

N/A

Generate calendar events

Name

Generate Calander Events

Goal

The purpose for this function is for the Saturday to add events, appointments, or tasks from their calendar, and offer a strategy on how to manage the calendar.

Input

Email analysis output

Output

The system will generate a visible entry on the user's calendar with a time allocation.

Main scenario

The user creates an entry manually using the application to ensure the time allocation of a specific event on their calendar.

Pre-conditions

Email must be processed by Saturday

Steps

1. Saturday will process the “email analysis” output.
2. Saturday will create a Calander event entry.

Post-conditions

A visible entry for the event is generated on the user's calendar with the specified time allocation.

Exceptional scenario

If there are scheduling conflicts or insufficient information provided, the system notifies the user and prompts them to provide additional details or choose an alternative time slot.

Analyze emails

Name

Analyze Emails

Goal

The purpose for this function is to allow the system to analyze events, appointments, or tasks

from their calendar to assess level of urgency, difficulty, time consumption and priority, to offer a strategy on how to better manage, complete and execute the calendar events.

Input

Incoming emails

Output

Saturday's analysis of the “email”.

Main scenario

The user receives emails, which the system will automatically process and assess. The system will generate the email analysis output which will serve as an input for the different functions (ie; Generate calendar events, create actionable email list .

Pre-conditions

1.User must be logged in.

2. User email must be logged in.

3. User must receive email.

Steps

Step1: User receives email.

Step2: Incoming email gets processed by NLP to allow the system to process and assess the content of the email.

Step3: the system will add the email to the actionable email list based on preset conditions (such as keywords, email addresses and email Titles) and the language/content of the email to evaluate priority level.

Step4: The system will create Calander entries to dedicate the time to execute the tasks based on the priority, urgency and time consumption of the task based on the time availability on the calendar.

Post-conditions

Generation of an output containing predetermined fields such as (Event name, Event description, Event duration, etc...)

Exceptional scenario

If there is insufficient information provided, the system notifies the user and

prompts them to provide additional details.

view emails

Name

View emails.

Goal

The purpose for this function is to allow users to access and manage their emails correspondence via the virtual assistant's analysis.

Input

User clicks on the email options.

Output

The system will display:

1. Organized summary of emails.
2. View unread and important emails.

Main scenario

A user wants to access their emails and make the content through the virtual personal assistant, to view the priority emails first and to get the valuable information highlighted.

Pre-conditions

1. The user needs to be logged in.
2. The user must link their email with the application.
3. The user needs to have an internet connection.

Steps

1. The user accesses the email option from the main page.
2. The virtual assistant will analyze the emails and categorize them based on priority.
3. The application will highlight the important items that needs to user attention.

Post- conditions

The user will be provided with an organized and categorized view of their emails.

Exceptional scenario

If the user email is no longer can be accessed by the application the system will:

1. Notify the user that the email can no longer be accessed by the system.

2. Suggest rechecking the email credentials.

3. Offer guidelines on how to reconnect email to the system.

view inbox <<extend>>

Name

View inbox.

Goal

The purpose for this function is to display to the user their inbox, while focusing on the messages that need more attention than the others.

Input

User clicks on the inbox button.

Output

The system will provide the user with their emails listed and categorized based on the priority.

Main scenario

A user wants to access their emails content while focusing on emails that require more attention.

Pre-conditions

1. The user needs to be logged in.

2. The user must link their email with the application.

3. The user needs to have an internet connection.

4. The emails must be analyzed by the system artificial intelligence.

Steps

1. the user clicks on the inbox selection.

2. The system displays the emails while keeping the user's attention on the more important emails.

Post- conditions

The user can manage their emails with the help of the virtual personal assistant AI.

Exceptional scenario

If the user email is no longer can be accessed by the application the system will:

1. Notify the user that the email can no longer be accessed by the system.
2. Suggest rechecking the email credentials.
3. Offer guidelines on how to reconnect email to the system.

view actionable emails <<extend>>

Name

View actionable emails.

Goal

The purpose for this function is to display to the user a list of emails that require action, such as deadlines, follow ups, tasks, and emails that still have not been read by the user.

Input

The user clicks on the actionable email's selection.

Output

The list of emails that the virtual assistant identified as actionable.

Main scenario

A user wants to instantly view and identify emails that have been analyzed and flagged as in need of action taking by the virtual personal assistant.

Pre-conditions

1. The user needs to be logged in.
2. The user must link their email with the application.
3. The user needs to have an internet connection.
4. The emails must be analyzed by the system artificial intelligence.

Steps

1. the user clicks on the actionable email's selection.
2. The system displays the emails that have been flagged important based on the analysis of the virtual personal assistant.

Post- conditions

Display of the list

Exceptional scenario

If the user email is no longer accessible by the system, the application will:

1. Notify the user that the email can no longer be accessed by the system.
2. Suggest rechecking the email credentials.
3. Offer guidelines on how to reconnect email to the system.

view to-do list <<extend>>

Name

View to-do list.

Goal

The purpose for this function is to display a to-do list for the user based on the actionable emails, providing the user with sub-tasks and reminders that will ease the process of compleϴng their daily tasks.

Input

User clicks on to-do list section.

Output

Organized agenda of tasks, items, and reminders based on the analysis of the virtual personal assistant.

Main scenario

The user wants to view a detailed list of actions that would aid them in simplifying their tasks and ensure that they have not missed a step in their process.

Pre-conditions

1. The user needs to be logged in.
2. The user must link their email with the application.
3. The user needs to have an internet connection.
4. The emails must be analyzed by the system artificial intelligence.

Steps

1. The user clicks on the to-do list feature.
2. The virtual assistant presents the to-do list, and agenda of tasks, items, and reminders

that the user needs to get their attention into.

Post- conditions

The user will have a comprehensive to-do list derived from the analysis of Saturday based on their emails, giving the user task prioritization and effective time management.

Exceptional scenario

If the user email is no longer can be accessed by the application the system will:

1. Notify the user that the email can no longer be accessed by the system.
2. Suggest rechecking the email credentials.
3. Offer guidelines on how to reconnect an email to the system.

Mark finished tasks <<extend>>

Name

Mark finished the tasks.

Goal

The purpose for this function is to allow the user to indicate the completion of a task, to aid the virtual assistant in its learning process to understand the user's attention and their appointed priority level to the task's category.

Input

1. The user clicks on the to do list.
2. The user selects the task that he/she wants to mark as completed.

Output

The system will update the to-do list for the user, marking the completed tasks, and keeping the user's attention on the pending tasks.

Main scenario

A user wants to confirm completion of tasks, seeking to keep the to-do list accurate and better manage future entries.

Pre-conditions

1. The user needs to be logged in.

2. The user must link their email with the application.

3. The user needs to have an internet connection.

4. The emails must be analyzed by Saturday.

Steps

1. the user opens the to-do list section.

2. The user selects and confirms the completed tasks.

3. The virtual personal assistant updates the to-do list according to the finished tasks.

4. The user views the updated to-do list.

Post- conditions

The to-do list is updated, reflecting the completed tasks.

Exceptional scenario

If the user email is no longer able to accessible by the system, the application will:

1. Notify the user that the email can no longer be accessed by the system.

2. Suggest rechecking the email credentials.

3. Offer guidelines on how to reconnect email to the system.

Non-functional requirements

1)Performance requirements :

- The duration of TIme it takes the system to respond to a user's service request is referred to as response TIme. In the case of our system Saturday responding to user requests should be as fast as possible, a good piece of sofware responds quickly because it can help the user more quickly.
- Implement tools to monitor the application's performance, track user interactions, and gather insights to make improvements while also being to constantly learn from the user's interaction.

2) Security and Safety requirements:

Whenever user data comes into play, the security and safety of said data always rests as the priority of the project's development focus. IdenTIfying all possible risks threatening the security and safety of the user’s data. Some of the requirements “Saturday” would need to meet to ensure the security and safety of said user data would be:

- Security Analysis: An in-depth invesTIgaTIon of “Saturday” security risks and threats should be completed and reported to ensure a maximized level of user data security, soFTware limitaTIons and risks.

- Security AdaptaTIon: “Saturdays” development team (“Team Omega”) must be able to adjust and improve the security features following the changes in the user’s needs.

- Security Framework: An appropriate security framework must be uTIlized with “Saturday” to ensure the connecTIvity of trusted networks only and the denial of any untrusted networks to prevent access of any unauthorized parties.

- Robustness: “Saturday” must be able to handle various unexpected user inputs.

- Error Handling: “Saturday” must be able to; detect, report, and resolve errors.

- Fault Tolerance: When designing the algorithms and code of “Saturday” the development team “Team Omega” must ensure a degree of failure and fault tolerance in the system and of its components, to avoid a domino effect of failures.

- Recovery: “Saturday” must be able to recover from system failures to ensure a quick and effecTIve recovery of failures and faults in data or systems.

High-level Design

System overview and design considerations

System Overview

Our system incorporates advanced Artificial Intelligence (AI) capabilities to enhance user productivity, particularly in managing tasks derived from emails. Leveraging the Mistral 7B AI model, we automate the extraction and organization of actionable tasks embedded within emails.

User Interface (React Native):
The user interface, developed using React Native, ensures a smooth and intuitive experience across iOS and Android platforms. Users can effortlessly interact with the application, viewing and managing tasks derived from their emails seamlessly.

Backend Services (Hosted on AWS EC2):
Our backend services, hosted on AWS EC2 instances, form the backbone of our system's functionality. These services encompass task processing, email parsing, and interaction with the Mistral 7B AI model. By harnessing the power of AWS EC2, we ensure scalability and reliability, vital for handling complex AI computations and managing user data securely.

AI Integration (Mistral 7B):
The centerpiece of our system's AI capabilities is the Mistral 7B model, renowned for its natural language processing (NLP) prowess. Integrated into our backend services, Mistral 7B analyzes incoming emails to identify and extract actionable tasks, transforming unstructured email content into organized task lists. This AI-driven approach streamlines task management, enabling users to prioritize and address important action items efficiently.

Database (SQLite):
To facilitate local data storage and offline access, we employ SQLite as our database solution. SQLite ensures rapid retrieval and storage of task-related information directly on users' devices, enabling seamless access to task lists even in offline environments.

Assumptions (T):

Our design approach is guided by several key assumptions that we made when conceptualizing our AI assistant app project. First of all, we expect that users will actively interact with the app, making use of its functions for calendar management, email filtering, and daily task management. Since the information handled by the app is sensitive, it is assumed that strict data privacy and security measures are essential. Furthermore, we expect the Natural Language Processing (NLP) algorithms to interpret user inputs accurately, resulting in a seamless and intuitive experience. Machine learning powers are needed for efficient email filtering and calendar management, provided that keywords and user preferences are correctly identified. To guarantee a consistent user experience, we also assume cross-platform compatibility. The user-friendly design of the app means that little to no user training is expected. Moreover, it is believed that regular maintenance and updates are essential for adjusting to changing industry standards and user needs. So, we conclude that these presumptions together serve as the foundation for our design choices, and ongoing evaluation and modification are essential to the project's success.

Constraints:

Our design choices for our AI assistant app project are informed by a foundational set of assumptions. First of all, we expect that users will actively interact with the program, taking advantage of its features to efficiently manage daily tasks, organize their calendars, and filter emails. This presumption stems from the conviction that users will find the app to have inherent value, which will encourage regular use. Because the information handled by the application is sensitive, we further assume that strict measures for data privacy and security are of utmost importance. Simultaneously, we foresee the need for precise Natural Language Processing (NLP) algorithms to ensure a smooth and intuitive user experience by accurately interpreting user inputs. The ability of machine learning algorithms to recognize keywords and understand user preferences is critical to the functioning of email filtering and calendar management. Furthermore, we believe that cross-platform compatibility is necessary to guarantee a consistent user experience on different devices. Because of the app's user-friendly design, we also expect that little to no user training will be needed. Finally, as a testament to our dedication to ongoing development, we believe that frequent updates and maintenance will be essential for adjusting to changing user demands and industry standards. (Experience, U. 2023) Even so, a number of limitations need to be recognized despite these presumptions. The main obstacles are the difficulty of achieving seamless cross-platform integration, the need for strong security measures, and potential limits in the accuracy of NLP algorithms. It is crucial to strike a balance between these assumptions and limitations to develop a robust design that not only meets user expectations but also tackles the problems that come with the ever-changing field of AI technology. Our approach will be refined through ongoing assessment and modification during the project's lifecycle, guaranteeing a design that not only meets but surpasses user expectations.

System environment :

When designing the system environment for our AI assistant app project, it is critical to carefully consider the wider ecosystem as well as the technology infrastructure. The application will mainly function in a cloud-based setting, taking advantage of the benefits of accessibility, scalability, and flexibility. Making use of cloud services like AWS or Azure will facilitate easy data processing, retrieval, and storage, creating an environment that supports the app's functionality and can grow as needed. Our project's main component, the integration of Natural Language Processing (NLP) algorithms, requires a complex computational environment. The complexity of NLP tasks will require high-performance servers with strong processing units in order to handle them and ensure quick and accurate user input interpretation. This computing power is essential to the responsiveness of the app and the general user experience. Furthermore, a properly configured machine learning environment is essential

because the app uses machine learning to recognize user preferences and filter emails.

Moreover, this idea includes specialized servers for model training and inference as well as frameworks like PyTorch and TensorFlow. To improve its functionality, the AI assistant app will interface with several external systems in addition to its technical infrastructure. Email filtering is not effective without a smooth integration with widely used email platforms such as Gmail and Outlook. Correspondingly, accurate scheduling and management of user events depend on integration with calendar systems like Google Calendar or Microsoft Outlook Calendar. To enable seamless communication between the application and external systems, these integrations will need strong Application Programming Interfaces (APIs) (Daniel M. Berry,2022). Moreover, the security environment in which the system functions is taken into account. Encryption protocols, secure socket layers (SSL), and multifactor authentication will be used to protect information because user data is sensitive. To find and fix any vulnerabilities, there will be regular security audits and ongoing monitoring. To sum up, our AI assistant app project's system environment is a complex web of cutting-edge technologies that includes cloud services, high-performance computing, machine learning frameworks, smooth integrations, and strong security measures. The foundation of the app's functionality, responsiveness, and general success in the rapidly changing field of artificial intelligence is this intricate and well-planned environment.

Design methodology:

A rigorous design process is essential when creating Saturday in order to guarantee the smooth incorporation of cutting-edge features while resolving any obstacles. In order to do that, we use the Double Diamond model as a guide and adopt an iterative design process. The first diamond requires in-depth analysis and investigation of user requirements and preferences. To collect both qualitative and quantitative data, we carry out extensive user interviews, surveys, and usability testing. We turn this data into actionable insights by using affinity mapping and user persona creation, which serve as the basis for our design choices. Moreover, we can better understand our target audience thanks to this user-centric approach, which guarantees that the app closely complies with their needs and expectations.

We use divergent thinking and ideation as we proceed into the second diamond. Design workshops and brainstorming sessions encourage creativity and produce a multitude of ideas. We visualize and hone these concepts using methods like wireframing, rapid prototyping, and user story mapping. The goal is to produce a thorough design blueprint that captures the essential features of the AI assistant application. Thorough usability testing is performed on every iteration to determine user satisfaction, and input is incorporated to improve the user experience even more.

In addition, our design process incorporates the Design Thinking framework's tenets, prioritizing ideation, empathy, and prototyping. A thorough grasp of user needs is the foundation for developing empathy, which helps us make sure that our designs suit the wide range of tastes of our target audience. In the process of coming up with novel ideas, cross-functional cooperation is essential. We can turn concepts into concrete versions quickly through rapid prototyping, which makes it easier to conduct early user testing and incorporate user feedback. The final AI assistant app will be creative and user-centric, meeting and surpassing the expectations of our diverse user base thanks to this iterative cycle, which is defined by a never-ending loop of design, testing, and refinement.

System architecture and use cases

System Architecture Design:

Presentation Layer:

Presenting the user interface of “Saturday”, which includes the main aspects of “Saturday’s” functionalities. Some of the components represented would be the calendar, populated with the users prescheduled events though out the time frame presented. The Inbox displays unopened un-opened email holding a value of high urgency according to the content. The To-Do list displays the list of actions to be done which would ensure the completion of the user’s overall task. The actionable email lists display the emails according to user settings which content indicate tasks delegated to the user from the sender of urgency or importance.

Application Layer:

This is the layer which contains the logical part of Saturday which can be broken down to simple steps:

(a) Email processing: The function responsible for the AI processing and analysis of incoming emails. This functionality is one of the driving forces of “Saturday” as it helps the user with automatic task generation and time allocation for said tasks which ensures the completion of the tasks.

(b) Generation of events: Post Email processing the AI module will be responsible to allocate the time of the user to tasks

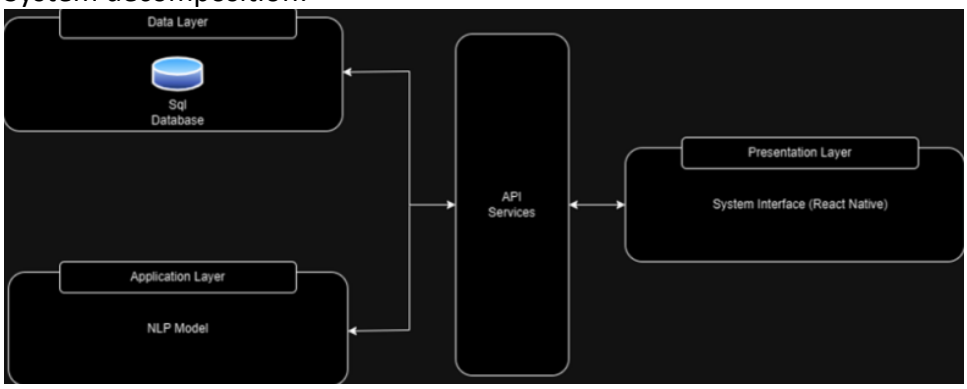
yet uncompleted. With the understanding of the user, the user's self-assessment of time consumption per task and access to the user's calendar finding optimal working hours to allocate for said task would be done automatically with the possibility of the user's intervention.

(c) To-Do list Generation: "Saturday's" functionality to generate a list of actions that help the user complete tasks by breaking them down into actions.

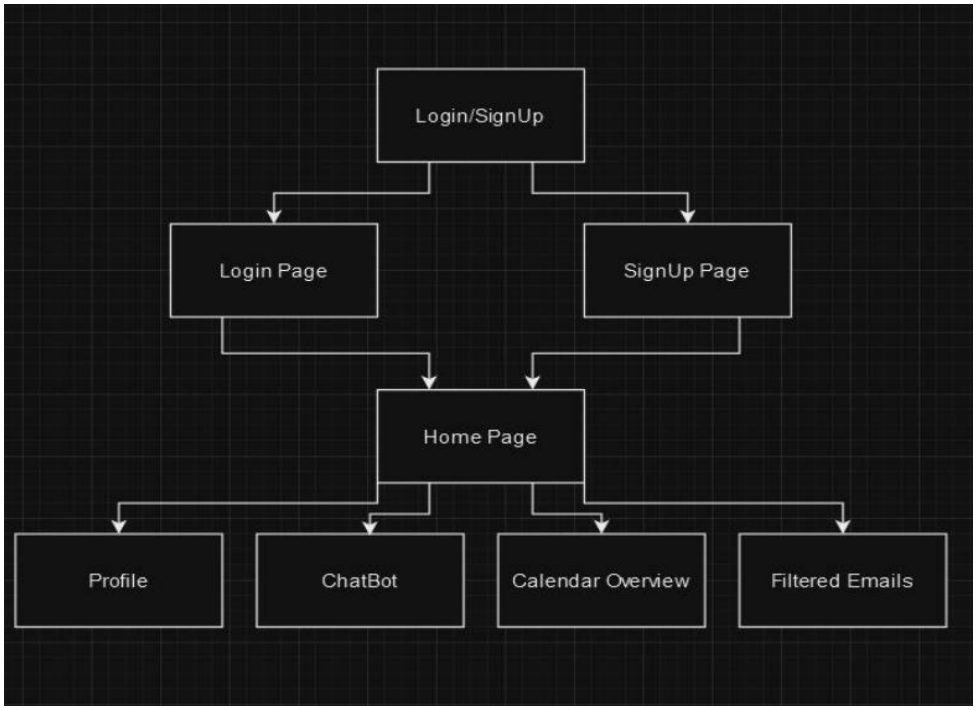
Data Layer:

Which is also known as database layer where the application data is stored and maintained. The data mentioned consists of User Id, VarChar() which holds the messages, and a Boolean which helps Identifying Identifying who sent the message whether it's Saturday or the user, created at time and updated at time.

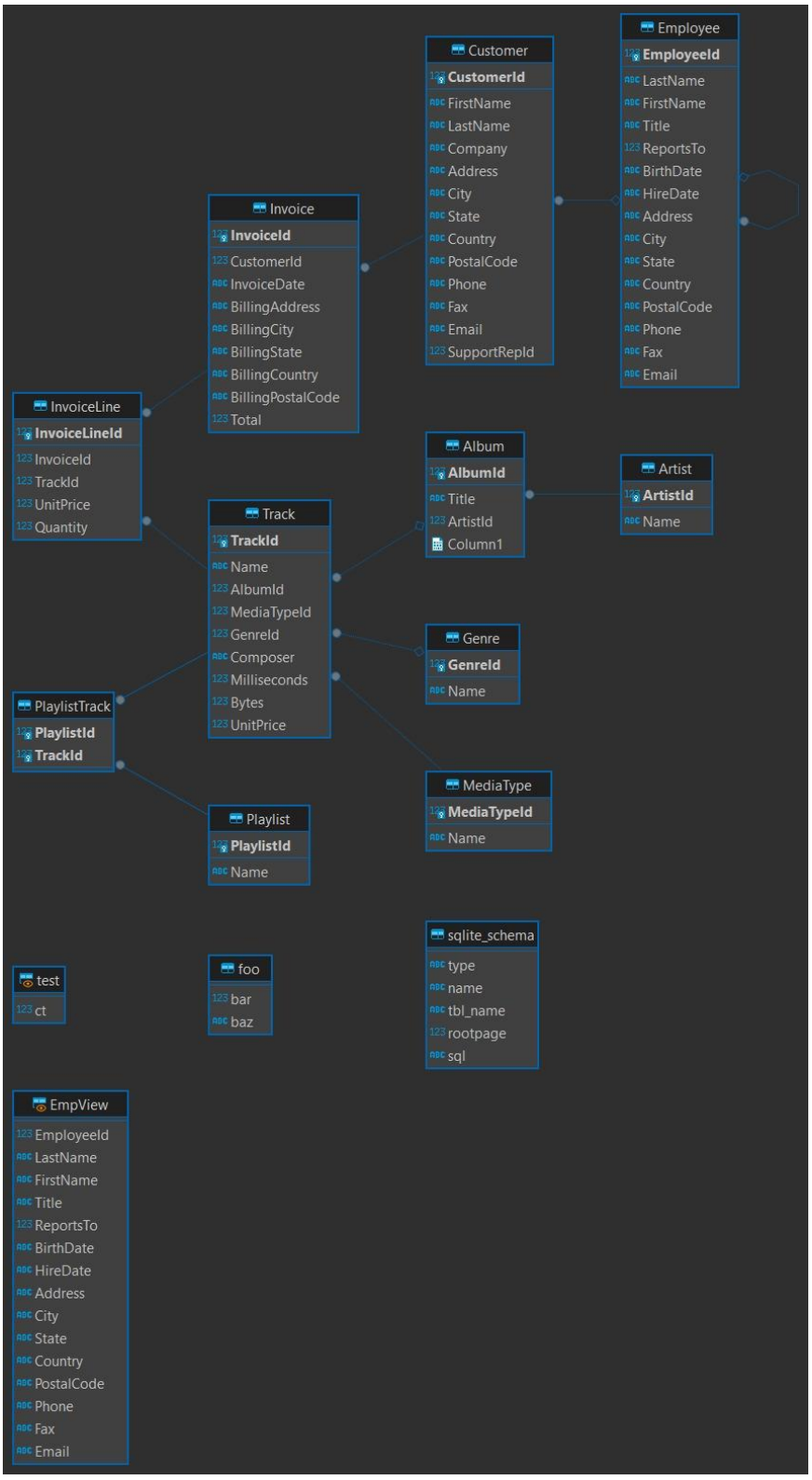
System decomposition:



Functional Decomposition Tree:

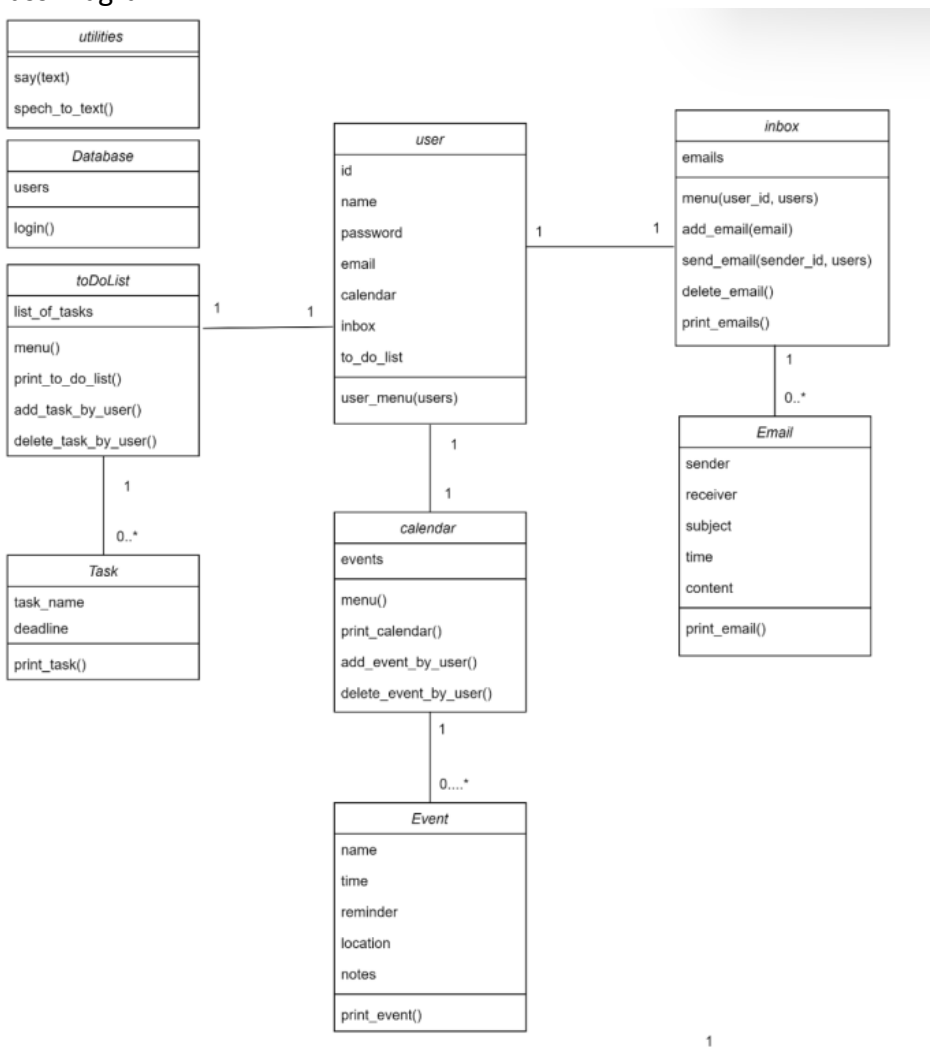

Login/SignUp
Login Page
SignUp Page
Home Page
Profile
ChatBot
Calendar Overview
Filtered Emails

ER diagram


Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId
Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email
Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total
InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity
Album
AlbumId
Title
ArtistId
Column1
Artist
ArtistId
Name
Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice
Genre
GenreId
Name
PlaylistTrack
PlaylistId
TrackId
MediaType
MediaTypeId
Name
Playlist
PlaylistId
Name
sqlite_schema
type
name
tbl_name
rootpage
sql
test
ct
foo
bar
baz
EmpView
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email

Class diagram

Class Diagram:



Class Breakdown:

Class **Utility** – this class contains functions that will allow users to say their commands and convert speech to text.

Class **User**– this class contains all the users and their credentials like login details.

Class **Inbox** – this class contains functions that would allow the user to interact with their inbox like composing, sending and deleting emails.

Class **Calendar** – the calendar class contains functions that would allow the user to manage their calendar. The user can add, delete and display events based on their preferences.

Class **Calendar** – the calendar class contains functions that would allow the user to manage their calendar. The user can add, delete and display events based on their preferences.

Class **Email** – this class displays all the emails the user has in their inbox.

Class **Event** – the event class displays all the events the user has in their calendar.

Class **To-Do List** – this class allows the user to manage their to-do list, by adding or deleting tasks. Moreover, it allows the user to view all the tasks in the list.

Class **Task** – this class displays all the tasks the user has on their calendar.

Class User	
Attributes	**Description**
Id	The user's unique ID identifier
Name	The user's username
Password	The user's password
Email	The user's email address
Calendar	The user's unique calendar
Inbox	The user's unique inbox
To_do_list	The user's unique To-Do list
Operations	**Description**
User_menu(users)	Returns the different menu options a user could access

Class Calendar	
Attributes	**Description**
Events	List that contains all the events that the user has in the calendar
Operations	**Description**
Menu()	Function will display different options to the user that they can perform in the

Class inbox	
Attributes	**Description**
Emails	List that contains all the user emails from the inbox class.
Operations	**Description**
Menu(user_id, users)	Function that displays multiple actions a user can perform on their inbox. (sending, deleting, or displaying emails)
Add_email(email)	Function that adds an email to the user's inbox **Param** an object of type email that contains (sender, receiver, subject, time, and content)
Send_email(sender_id, users)	Function allows the user to send an email to any other user. **Param** the ID of the sender and the list of the users in the system that are stored in the database class
Delete_email()	Function that allows the user to access the inbox and delete an email. **Throws** Invalid email number

Display_email()	Function that allows the user to view their inbox, and display all emails

	calendar. (Display events, add or delete an event)
display_calendar()	The function will display all the events on the calendar to the user.
Add_event_by_user()	Function that will allow the user to add an event to the calendar.
Delete_event_by_user()	Function that will allow the user to delete an event from the calendar. **Throws** Invalid event number

Class Event	
Attributes	**Description**
Name	Name of the event
Time	The time that the event will occur
Reminder	The number of reminders the user wants
Location	Location of the event
Notes	Additional notes the user wishes to add to the event
Operations	**Description**
Display_event ()	Function that displays all the information about a specific event.

Class Email	
Attributes	**Description**
Sender	The sender's email address
Receiver	The receiver's email address
Subject	The subject of the email
Time	Timestamp of the email sent
Content	Content of the email
Operations	**Description**
Display_email()	Function that displays the user email

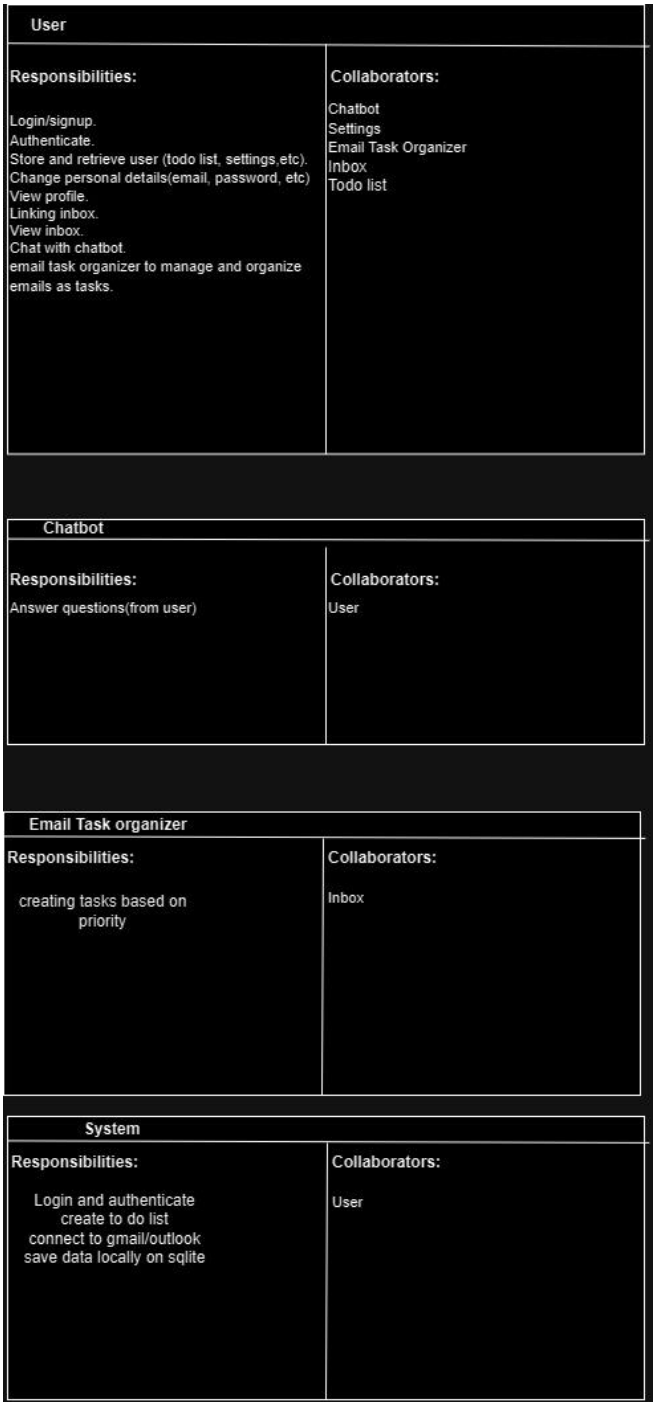
Class Todo List	
Attributes	**Description**
List_of_tasks	List that contains all the user's tasks.
Operations	**Description**
Menu()	Function that displays different options a user can perform on their list. (Viewing the list, adding, or deleting a task)
Display_to_do_list()	Function that displays all the tasks in the to-do list.
Add_task_by_user()	Function that allows the user to access the to do list and add tasks.
Delete_task_by_user()	Function that allows the user to access the to do list and delete tasks. **Throws** Invalid task number

Class Task	
Attributes	**Description**
Task_name	Name of the task
Deadline	The deadline the user will have to finish the task.
Operations	**Description**
Display_task()	Function that displays information of the task.

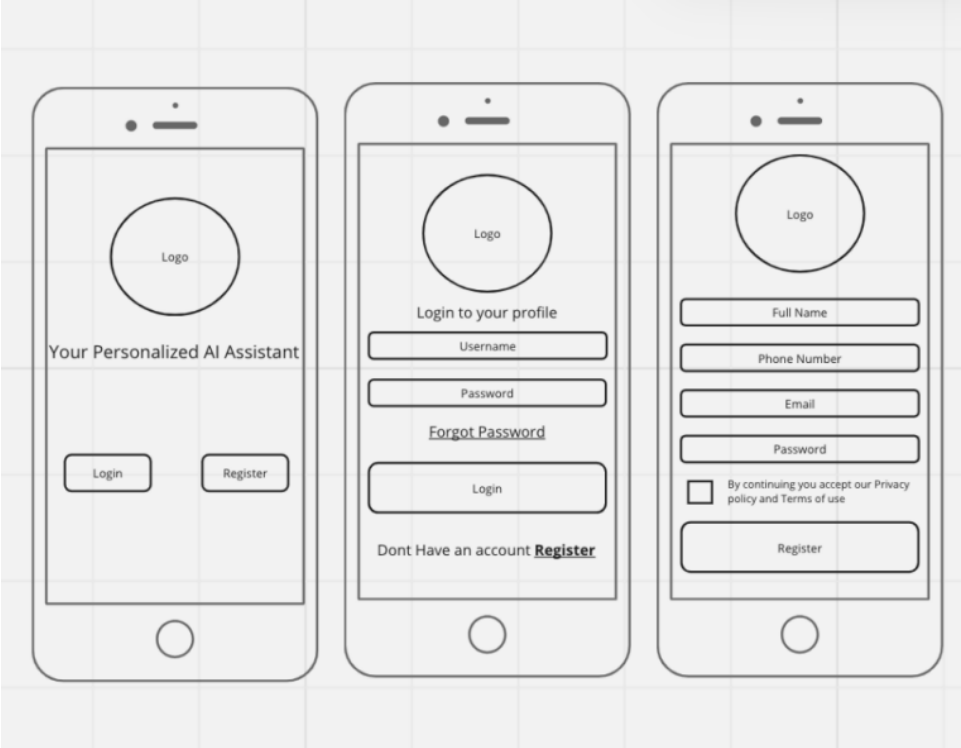
Say(text)	
Speech_to_text()	Function that allows the user to say the commands instead of writing. **Throws** sr.UnknownValueError and sr.RequestError.

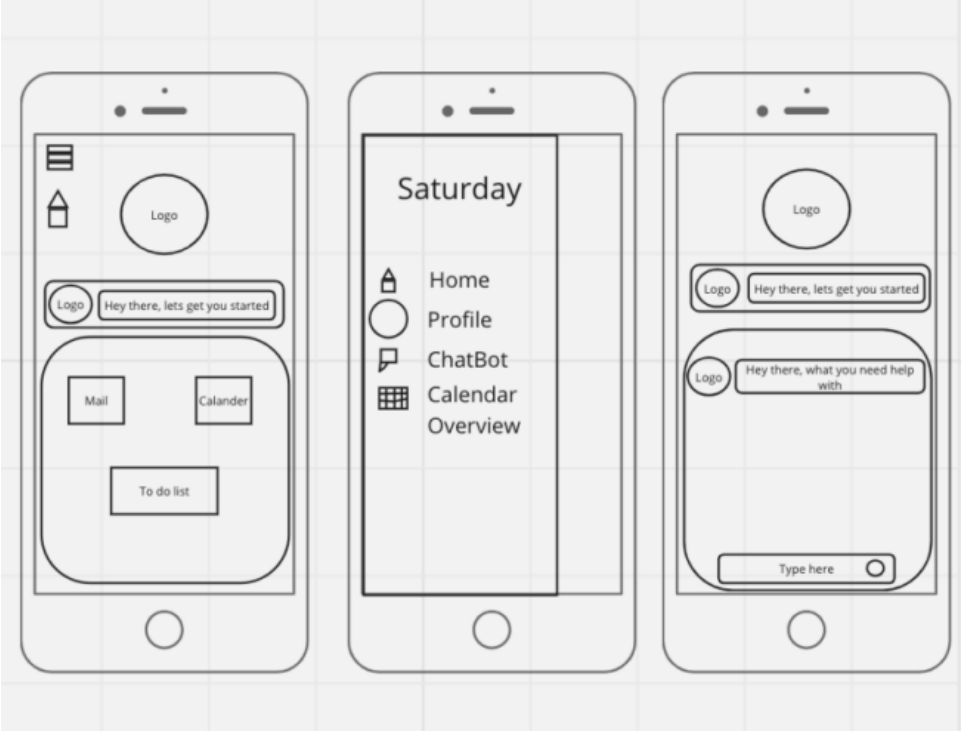
Class Database	
Attributes	**Description**
Users	List that contains all the users.
Operations	**Description**
Login()	Function that validates the user's credentials and grants them access. **Return** None

CRC cards

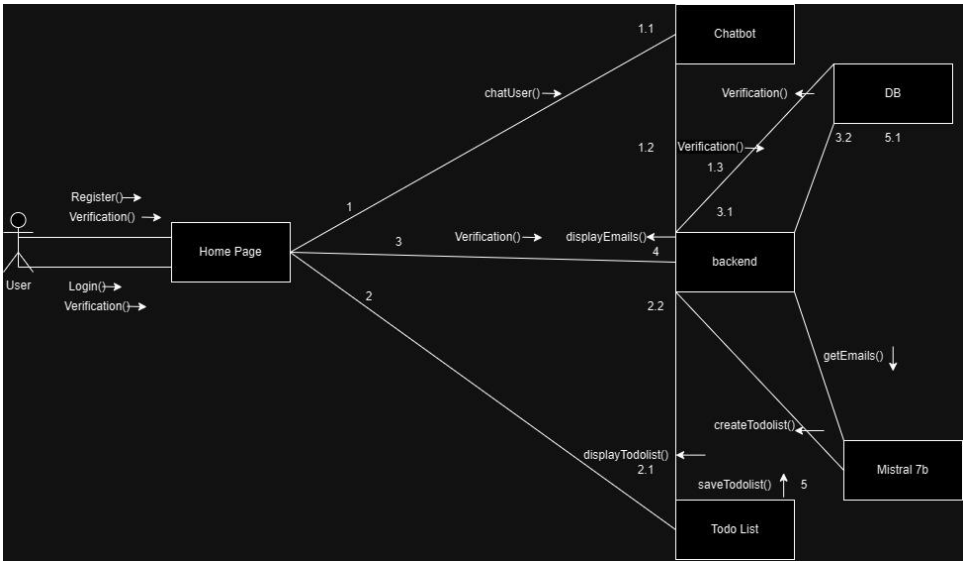

User
Responsibilities:
Login/signup.
Authenticate.
Store and retrieve user (todo list, settings,etc).
Change personal details(email, password, etc)
View profile.
Linking inbox.
View inbox.
Chat with chatbot.
email task organizer to manage and organize emails as tasks.
Collaborators:
Chatbot
Settings
Email Task Organizer
Inbox
Todo list
Chatbot
Responsibilities:
Answer questions(from user)
Collaborators:
User
Email Task organizer
Responsibilities:
creating tasks based on priority
Collaborators:
Inbox
System
Responsibilities:
Login and authenticate
create to do list
connect to gmail/outlook
save data locally on sqlite
Collaborators:
User

User interfaces





Collaboration diagram



Low-level Design

Pseudo codes

Due to the large number of functions which make up “Saturday”, creating a pseudo code for each function of “Saturday’s” three aspects (Front End, Back End & AI) would be infeasible. Therefore, to still provide as much information on the low-level design of “Saturday’s” codes and development, each department of the team has selected one main feature of the three aspects to write the pseudo code for and present in this report. Further documentation could be provided upon request. Due to the variety of formats when writing pseudo codes, we have decided upon using a mixture of formats to ensure maximum coverage of the various formats.

Front End:

Email screen pseudocode

```
Component Index {
    Initialize emails as empty array
    Initialize selectedEmailID as null
    Use navigation for screen transitions
    Get sessionId from global state

    On component mount {
      Define asynchronous function fetchEmailsData:
        Try to fetch emails using sessionId
        Set fetched emails to state
        Catch and log any errors

      Call fetchEmailsData
    }

    Define handleEmailPress function taking EmailID:
      Update selectedEmailID state with passed EmailID

    Component Render:
      Display a container view
        Display a title 'Email Highlights'
        Display an image for email
        Display a list container
          For each email in emails:
            Display an email item
              If item.EmailID equals selectedEmailID, highlight it
              Display sender and subject of the email
              Handle press on the email item to call handleEmailPress
  }
```

Back End:

```
class ExternalServiceView(APIView):
   permission_classes = [IsAuthenticated]
   serializer_class = MessageSerializer

   function post(self, request, *args, **kwargs):
      // Translate the input message to appropriate format
      translate_message(data=request.data)
      // Validate the translated message
      if valid_message():
         // Process the message if user is authenticated
         if user_authenticated(request.user):
            create_chatbot_entry(UserID=request.user,
message=message_data['message'], isUser=True)
         else:
            return UnauthorizedError("User must be authenticated to send
messages.")
         try:
            // Send the message to external service
            response =
send_message_to_external_service(url="http://52.63.121.96/api/chat",
data=message_data)
            // Check if message sent successfully
            if response.status_code == 201:
               // Process the response from external service
               process_response(response)
               return SuccessResponse(response.json())
            else:
               return ErrorResponse("External service error",
response.status_code)
         except ExternalServiceError as e:
            return ServiceUnavailableError(str(e))
      else:
         // Return errors if message validation fails
         return BadRequestError(serializer.errors)
      // End of post function
```

AI:

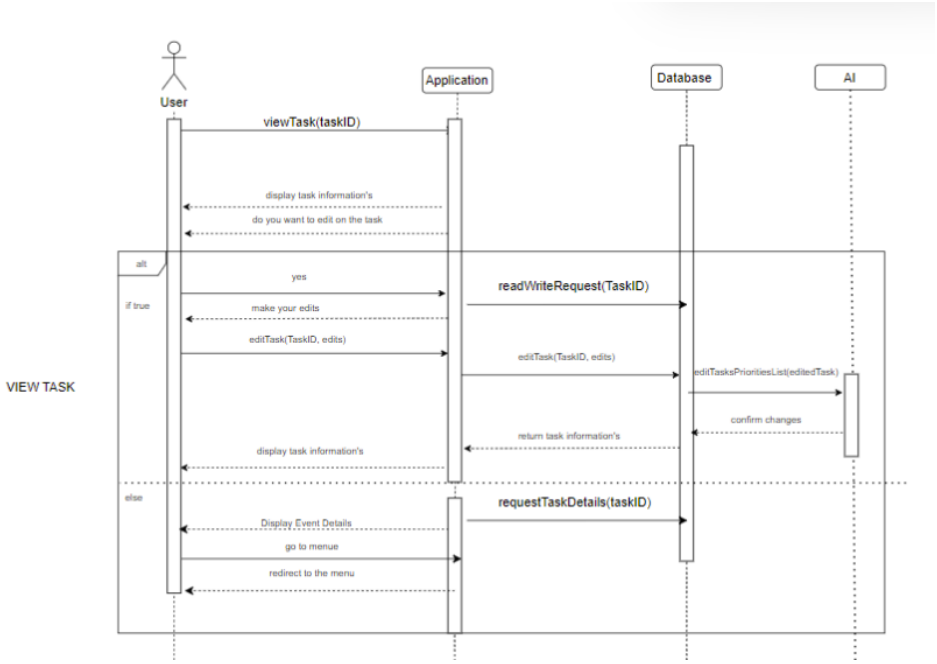
```
FUNCTION generate(prompts: List[str], model: Transformer, tokenizer: Tokenizer, *,
max_tokens: int, temperature: float, chunk_size: int = None):
   model = model.eval()
   B, V = length of prompts, model's vocabulary size
```

```
encoded_prompts = []
FOR EACH prompt IN prompts:
    Encode prompt using tokenizer and append to encoded_prompts

seqlens = []
FOR EACH encoded_prompt IN encoded_prompts:
    Calculate length of each encoded prompt and append to seqlens

cache_window = maximum(seqlens) + max_tokens
IF model's sliding_window is not None and cache_window > model's sliding_window:
    cache_window = model's sliding_window

Initialize cache as RotatingBufferCache with specified parameters
Reset cache

Initialize logprobs as a list of lists with B empty lists
Initialize last_token_prelogits as None

IF chunk_size is not specified:
    chunk_size = maximum length among seqlens

FOR s FROM 0 TO maximum length among seqlens WITH step size chunk_size:
    Divide encoded_prompts into chunks of size chunk_size

    prelogits = model's forward pass with concatenated prompt chunks, seqlens, and cache

    Calculate logits by applying log_softmax on prelogits

    IF last_token_prelogits is not None:
        Calculate log probabilities for previous tokens

    Update logprobs with log probabilities of current tokens

    Update last_token_prelogits with prelogits of last tokens

generated_tokens = []
ASSERT last_token_prelogits is not None

FOR i_token FROM 0 TO max_tokens:
    Generate next token using sampling method

    Calculate log probabilities for generated token
```

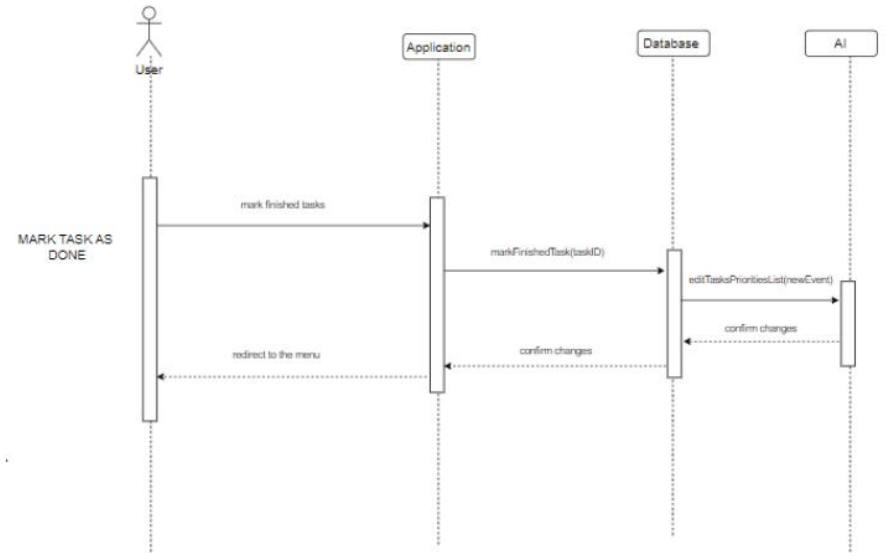
```
        Append next_token to generated_tokens

        Update last_token_prelogits with model's forward pass for next_token and cache

    Decode generated_tokens into words using tokenizer
    RETURN generated_words, logprobs

FUNCTION interactive(model_path: str, max_tokens: int = 35, temperature: float = 0.7, instruct:
bool = False):
    Initialize tokenizer with path to tokenizer model
    Initialize transformer from folder with specified max_batch_size

    WHILE True:
        Prompt user for input
        IF instruct is True:
            Prepend and append prompt with instructional tokens

        Generate response using generate function

        Print generated response
```

Sequence diagrams

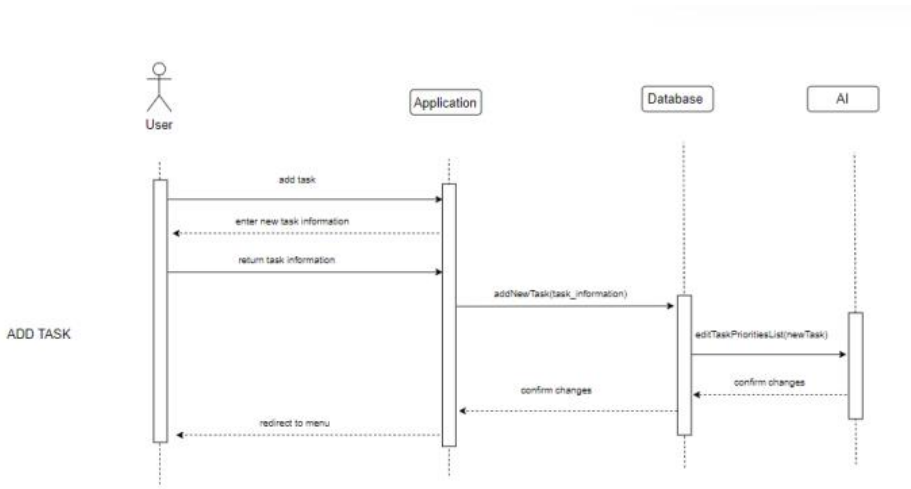
View Task:



Mark Task As Done:



Add Task:



Implementation details

Our implementation utilizes React Native for the user interface, offering a native-like experience across platforms. React Native components are designed to ensure smooth user interaction. Meanwhile, our backend infrastructure, encompassing both traditional backend functionalities and AI processing, is hosted on AWS EC2 instances, benefiting from their scalability and flexibility. We've integrated AWS EC2 for hosting our backend application and AI services, ensuring reliable performance and seamless scalability to meet varying demands. Additionally, for database management, we've opted for SQLite, providing lightweight storage directly on the user's device for offline access and efficient data retrieval. This architecture not only ensures a responsive and intuitive user experience but also leverages the robust capabilities of AWS EC2 for backend operations.

Testing

Testing goals

A key component of every innovation's success is maintaining product integrity. This in-depth analysis from Team Omega explores the core of our most recent project, "Saturday," carefully analyzing the stability of our product's base. We next dive into an in-depth analysis of "Saturday's" backend features, emphasizing the key elements of each function: its goal, necessary conditions, input specifications, and the results under both favorable and unfavorable circumstances. Our goal is to demonstrate how "Saturday" has been precisely designed to manage a variety of scenarios while maintaining a smooth operational flow. It's crucial to keep in mind that certain features are still being refined and have not yet reached their full potential as we move through the development process. This document promises to maintain a constant focus on

quality and dependability as we move forward, demonstrating our dedication to excellence and the continuous improvement of our product.

Test plan scope

By the end of the testing procedure. The team shall identify the issues, breakdowns and gaps in the codes run-time integrity when faced with unplanned inputs. Form ensuring proper error handling, to ensuring the completion of the back-end functionalities, increasing the code security, and proper handling of errors when arise.

Test forms and test results

Test form:

The procedure of recording the testing and the testing method conducted in this report was a fully manual acceptance series of tests for all the backend functions.

To ensure a thorough overview of the back-end development, team Omega's back-end development division iterated over every function individually and manually to review, test and document each function. To ensure the integrity of the testing result, the work was peer-reviewed by the division members, then shared with the entire team for further verification of work and validation of the result.

Within the testing document there exists 48 acceptance tests for each function of "Saturday's" backend. Please refer to the "TEST PLANNING" Document for further information.

Sample of testing procedure:

Function 1

Requirement name:	User Registration
Test case description:	The user enters relevant and necessary information to register an account.
Pre-requirements:	Internet connection to the application servers.
Input:	User navigates to the registration page and inputs their credentials. Email, Username, Name, Phone number, Profession, Password.
Expected output (positive):	Verification of entered data. Successful message referring user credentials saved. User redirected to the OTP.
Expected output (negative):	“Specified” filed incorrect or doesn’t meet the requirements.
Observed output:	The user is able to register their credentials into the database and is redirected to the OTP screen to verify their account.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 2

Requirement name:	OTP verification.
Test case description:	Verifying valid email address ownership to new users. By passing the OTP code sent to the specified email.
Pre-requirements:	Internet connection to the application server valid entry of registration information access to the specified email account.
Input:	The user must enter the OTP code received on their private email into the verification prompt.
Expected output (Positive):	” Account created successfully” notification prompt
Expected output (Negative):	” Invalid OTP” notification prompt
Observed output:	The user is able to successfully register and is redirected to the login page.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 3

Requirement name:	User profile view.
Test case description:	Display of account information to user.

Pre-requirements:	Internet connection to application server User logged in to his/her account.
Input:	User navigates to the profile page.
Expected output (positive):	The function returns the user's information.
Expected output (Negative):	"User is not authenticated" notification message. Redirecting to user login page.
Observed output:	The user is able to view their information.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 4

Requirement name:	Change Password
Test case description:	Allows a logged-in user to change account password.

Pre-requirements:	Internet connection to the application server Successful login to the registered account.
Input:	The user must enter the current password & a valid new password.
Expected output (Positive):	“Password updated successfully” notification message.
Expected output (Negative):	“Wrong current password” notification message. The new password doesn’t meet the requirements.
Observed output.	The user is able to change their password and is redirected to the profile page.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 5

Requirement name:	Change of Email.
Test case description:	User attempts to change the Email registered to their account.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.

Input:	Users navigate to the profile page, then enters the new email address.
Expected output (Positive):	“Email updated Successfully” notification message. New email doesn’t meet the requirements such as “@” not present in the email address.
Expected output (Negative):	“Invalid Email Format” notification message.
Observed output:	The user is able to change their email address and is redirected to the profile page.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 6

Requirement name:	Change Profession
Test case description:	User attempts to change the profession set in their profile description.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.

Input:	User navigates to the profile page, then enters the new profession.
Expected output (Positive):	“Profession updated successfully” notification update.
Expected output (Negative):	-
Observed output:	The user is able to change their profession is redirected to the profile page.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 7

Requirement name:	Change Phone Number
Test case description:	The user attempts to change the registered phone number on the account.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.

Input:	Users navigates to the profile page, then enters the new phone number.
Expected output (Positive):	“Phone updated successfully” notification update.
Expected output (Negative):	Phone number does not meet the requirements such as specific length, nothing other than integer sent.
Observed output:	The user is able to change their phone number and is redirected to the profile page.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 8

Requirement name:	Email Statistics.
Test case description:	Returns the statistics of opened and unopened emails of the day.
Pre-requirements:	ernet connection to application server er logged in to the account

Input:	The application automatically fetches the user's statistics of opened/unopened emails.
Expected output (positive):	Unopened emails of the today. Opened emails of the today. Total emails of the today.
Expected output (Negative):	-
Observed output:	The user is able to view their progress of reading emails of the day.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Calendar Section

Function 9

Requirement name:	View Calendar Events/Tasks

Test case description:	User attempts to view Calendar events and to-do tasks.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.
Input:	User navigates to the home page and clicks on the calendar.
Expected output (Positive):	Display details of entries such as: (Event ID, Title, Event Description, Start Date, Start Time, End Date, End Time, Destination, Is Deleted, created at, updated at, and User ID)
Expected output (Negative):	-
Observed output:	The user is able to view their events.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 10

Requirement name:	Specific Event View

Test case description:	The user attempts to view the details of a specific entry
Pre-requirements:	ernet connection to the application server ccessful login to the registered account sting of Event entry.
Input:	User navigates to the Calendar page and selects the event to be viewed.
Expected output (Positive):	Display details of entry such as: (Event ID, Title, Event Description, Start Date, Start Time, End Date, End Time, Destination, Is Deleted, created at, updated at, and User ID)
Expected output (Negative):	“Event Item not found” notification message
Observed output:	The user is able to view all the details of an event.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 11

Requirement name:	Create Entry

Test case description:	The user attempts to create a new calendar/task entry
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.
Input:	User navigates to calendar page and enters details such as: Title, Event Description, Start Date, Start Time, End Date, End Time, and Destination.
Expected output (Positive) :	*Event Created successfully* notification
Expected output (Negative) :	“Invalid * value” notification message
Observed output:	The user is able to create a new event which is displayed onto their calendar.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 12

Requirement name:	Delete Event Entry
Test case description:	The user attempts to delete a calendar/task entry.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account sting event
Input:	User navigates to the calendar page, selects the event, and then delete its.
Expected output (Positive):	“Entry Deleted” notification.
Expected output (Negative):	“Event item not found” notification.
Observed output:	The user is able to delete the event from the calendar and the updated calendar is displayed.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 13

Requirement name:	Change Entry Title
Test case description:	The user attempts to change the title of a calendar entry.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account sting event
Input:	User clicks on the Calendar, selects the specific event, then enters a new title.
Expected output (Positive):	"Event Title Updated successfully" notification.
Expected output (Negative):	"Event item not found" notification if the user is trying to change an event that doesn't belong to them.
Observed output:	The user is able to change the event's title. The updated title is displayed on the user's calendar.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 14

Requirement name:	Change Entry Description
Test case description:	The user attempts to change the description of a specific calendar entry.
Pre-requirements:	ernet connection to the application server, ccessful login to the registered account sting event
Input:	User clicks on the calendar, selects a specific event, then enters the new description.
Expected output (Positive):	"Event Description Updated successfully" notification.
Expected output (Negative):	"Event item not found" notification if the user is trying to change an event that doesn't belong to him.
Observed output:	The user is able to change the event's description. The new description is displayed onto the calendar.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 15

Requirement name:	Change Entry Start Date
Test case description:	The user attempts to change the start date of a specific calendar entry.
Pre-requirements:	ernet connection to the application server. ccessfully logged into the registered account. existing event entry.
Input:	User clicks on the calendar, then the specific event, and lastly enters the new start date.
Expected output (Positive):	“Event Start Date Updated Successfully” notification.
Expected output (Negative):	“Event Item not found” notification if the user is trying to change an event that doesn’t belong to him. Invalid date format.
Observed output:	The user is able to change the entry’s start date. The updated date is displayed on the user’s calendar.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 16

Requirement name:	Change Entry Start Time
Test case description:	The user attempts to change the start time of a specific calendar entry.
Pre-requirements:	ernet connection to the application server. ccessfully logged into the registered account. existing event entry.
Input:	User clicks on the calendar, then the specific event, and lastly enters the new start time.
Expected output (Positive):	“Event Start Time Updated Successfully” notification.
Expected output (Negative):	“Event Item not found” notification if the user is trying to change an event that doesn’t belong to him. Invalid time format.
Observed output:	The user is able to change the entry’s starting time. The updated time is displayed on the user’s calendar.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 17

Requirement name:	Change Entry End Date
Test case description:	The user attempts to change the end date of a specified calendar entry.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account sting event
Input:	User clicks on the calendar, selects a specific event, then enters the new end date.
Expected output (Positive):	"Event End Time Updated Successfully" notification.
Expected output (Negative):	"Event item not found" notification if the user is trying to change an event that doesn't belong to him. Invalid date format.
Observed output:	The user is able to change the entry's ending time. The updated time is displayed on the user's calendar.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 18

Requirement name:	Change Entry End Time
Test case description:	The user attempts to change the end date of a specified calendar entry.
Pre-requirements:	ernet connection to the application server. ccessfully logged into the registered account. d existence of Event entry.
Input:	User clicks on the calendar, then the specific event, and lastly enters the new end time.
Expected output (Positive):	“Event End Time Updated Successfully” notification.
Expected output (Negative):	“Event item not found” notification if the user is trying to change an event that doesn’t belong to him. Invalid time format.
Observed output:	The user is able to change the entry’s ending time. The updated time is displayed on the user’s calendar.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 19

Requirement name:	Event Duration Check
Test case description:	The users attempts to check the duration of the travel time from the location of one event to the other using Google Maps' APIs.
Pre-requirements:	ernet connection to the application server. ccessfully logged into the registered account. d existence of two Event entries with pre-specified location per event.
Input:	The system automatically checks for the duration when a user wishes to add an event.
Expected output (positive):	Duration of the trip in minutes.
Expected output (Negative):	"One or both events not found" notification. Invalid travel method.
Observed output:	System allows the user to add an event.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 20

Requirement name:	Email View
Test case description:	The user attempts to refresh the inbox to retrieve new emails from the email inbox linked with the user account.
Pre-requirements:	ernet connection to the application server. ccessful login to the registered account.
Input:	Navigate to the email inbox page and scroll down to refresh.
Expected output (Positive):	Returns all new emails from the application database and new emails might be present in the email inbox liked with the user account in Saturday application.
Expected output (Negative):	Error message based on the error type {“Connection error”}
Observed output:	The user is able to view their inbox saved in the application database and fetch new emails in the email inbox.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 21

Requirement name:	Inbox refresh
Test case description:	The user attempts to view the email inbox (from the user database), without fetching new emails that might be present in the user inbox linked to the account.
Pre-requirements:	ernet connection to the application server. ccessful login to the registered account.
Input:	Navigate to the email inbox page.
Expected output (Positive):	Presents all emails of the user saved in the database
Expected output (Negative):	-
Observed output:	The user is able to view all emails saved in the database of Saturday.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 22

Requirement name:	Delete Email
Test case description:	The user attempts to delete an email from the application database.
Pre-requirements:	ernet connection to the application server. ccessfully logged into the registered account. existing email.
Input:	User clicks on email, selects the message to be delete, and then clicks on the delete button.
Expected output (Positive):	“Email item deleted successfully” notification.
Email output (Negative):	“Email item not found” notification.
Observed output:	The user is able to delete a specific email from Saturday’s database.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 23

Requirement name:	Specific email view
Test case description:	The user attempts to view details of a specific email from their inbox.
Pre-requirements:	ernet connection to the application server. ccessfully logged into the registered account. existing email.
Input:	User clicks on the inbox page, then clicks on the email to view it's details.
Expected output (Positive):	Display of email details such as (Email ID, Sender, Receiver, Subject, ReceivedDate, IsPriority, IsDeleted, IsOpen, Body, UserID).
Expected output (Negative):	"Event item not found" notification.
Observed output:	The user is able to view all the details of a specific email message.

Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 24

Requirement name:	Remove email address from priority list.
Test case description:	The user attempts to remove a sender's address from the priority email addresses.
Pre-requirements:	ernet connection to the application server. ccessfully logged into the registered account. existing email address entry.
Input:	User clicks on the Settings page, selects the Priority List option, selects the required email address and deletes the entry.
Expected output (Positive):	"Priority email deleted successfully" notification.
Expected output (Negative):	"Priority email not found" notification.

Observed output:	The user is able to delete an email address from the Priority List. Then the list is updated.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 25

Requirement name:	Add Priority Email Addresses
Test case description:	User attempts to add an email address to the priority addresses.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.
Input:	User clicks on the Settings page, selects the Priority List option, then add a new priority email address.
Expected output (Positive):	“Email address added to priority successfully” notification.
Expected output (Negative):	Email address doesn’t meet the requirements.
Observed output:	User successfully added an email address to the priority list, and the added account is visible in the email priority list.

Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 26

Requirement name	View priority email address list.
Test case description:	User attempts to view the list of email addresses set as priority.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.
Input:	User clicks on the Settings page, selects the Priority List option to view the email addresses.
Expected output (Positive):	Returns list of priority email addresses.
Expected output (Negative):	-
Observed output:	User successfully views the emails that they set as priority.

Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 27

Requirement name:	Change email priority status.
Test case description:	User wants to change the IsPriority status of an email.
Pre-requirements:	ernet connection to the application server. ccessfully logged into the registered account. existing email entry.
Input:	User clicks in the inbox page, selects the email message, and changes it's priority accordingly.
Expected output (Positive):	"Email priority updated successfully" notification.

Expected output (Negative):	“Email not found” notification.
Observed output:	The user is able to change the priority of a specific email.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 28

Requirement name:	Event Statistics
Test case description:	Returns the statistics of previous and future events of the day.
Pre-requirements:	ernet connection to application server er logged in to the account.
Input:	The application automatically fetches the user’s statistics of previous and future events.

Expected output (positive):	Previous events of the day. Future events for the day.
Expected output (Negative):	-
Observed output:	The user is able to view each day's progress of completed tasks.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

To-Do List Section

Function 29

Requirement name:	View To-Do List Tasks
Test case description:	User attempts to view TDL events and to-do tasks.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.
Input:	User clicks on the To-Do-List page to view all tasks.
Expected output (Positive):	Display details of entries such as: (Event ID, Reminder, priority, due date, description, category, notes, status)
Expected output (Negative):	

Observed output:	The user is able to view their tasks.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 30

Requirement name:	Viewing a specific To-Do Event
Test case description:	The user attempts to view the details of a specific entry
Pre-requirements:	ernet connection to the application server ccessfully logged into the registered account. existing event entry
Input:	User clicks on the To-Do-List page to view all tasks, then selects a task to be viewed.
Expected output (Positive):	Display details of entry such as (Event ID, Reminder, priority, due date, description, category, notes, status)
Expected output (Negative):	“Event Item not found” notification message

Observed output:	The user is able to view all the details of a task.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 31

Requirement name:	Create To-Do List Entry
Test case description:	The user attempts to create a new entry for the To-Do list.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account.
Input:	User clicks on the To-do List page, adds appropriate task entry details. (Reminder, priority, due date, description, category, notes, status values.)
Expected output (Positive):	*Event Created successfully* notification
Expected output (Negative):	“Invalid * value” notification message
Observed output:	The user is able to add a new task to their To-do List.

Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 32

Requirement name:	Delete To-Do Event Entry
Test case description:	The user attempts to delete a task entry from the To-do List.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account existing event entry
Input:	User clicks on the To-do List page, selects the task and deletes it.
Expected output (Positive):	“Entry Deleted” notification.
Expected output (Negative):	“Event item not found” notification.
Observed output:	The user is able to delete a task from the To-do List. The list is updated and displayed to the user.

Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 33

Requirement name:	Change To-Do Entry Reminder
Test case description:	The user attempts to change the title of a reminder in the list.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account existing event entry
Input:	The user clicks to the To-do List, selects a task, and changes it's date/time values.
Expected output (Positive):	"Event Reminder Updated successfully" notification.
Expected output (Negative):	"Event item not found" notification. Wrong reminder date format.

Observed output:	The user is able to change a task's reminder date. The updated date is displayed on the user's To-do List.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 34

Requirement name:	Change Entry Description
Test case description:	The user attempts to change the description of a specified calendar entry.
Pre-requirements:	ernet connection to the application server, ccessful login to the registered account existing event entry
Input:	The user clicks on the To-do List, selects a task and changes it's description.
Expected output (Positive):	"Event Description Updated successfully" notification.
Expected output (Negative):	"Event item not found" notification.

Observed output:	The user is able to change a task's description. The updated description is displayed to the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 35

Requirement name:	Change To-Do Entry Priority
Test case description:	The user attempts to change the Priority of a specific entry from the list.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account existing event entry
Input:	The user clicks on the To-do-list, selects a task, and enters the new task priority.
Expected output (Positive):	"Event Priority Updated Successfully" notification.

Expected output (Negative):	“Event Item not found” notification. Wrong priority value format.
Observed output:	The user is able to change the priority of an entry. The updated priority is displayed to the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 36

Requirement name:	Change To-Do Entry Due Date
Test case description:	The user attempts to change the due date of a specific calendar entry.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account existing event entry
Input:	The user clicks on the To-do List, selects a task, and enters the new due date.
Expected output (Positive):	“Event Due Date Updated Successfully” notification.

Expected output (Negative):	“Event Item not found” notification. Wrong date format. Due date format is wrong.
Observed output:	The user is able to change the due data of a specific entry. The updated entry is displayed to the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 37

Requirement name:	Change To-Do Entry Category
Test case description:	The user attempts to change the category of a specific entry from the To-do List.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account existing event entry
Input:	User clicks on the To-do List, selects a task, and enters the new category for the task.
Expected output (Positive):	“Event Category Updated Successfully” notification.

Expected output (Negative):	“Event item not found” notification.
Observed output:	The user is able to change the category of the task. The updated task is displayed to the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 38

Requirement name:	Change To-Do Entry Notes.
Test case description:	The user attempts to change the notes of an entry from the To-do List.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account existing event entry
Input:	User clicks on the To-do List, selects a task, and changes the notes accordingly.
Expected output (Positive):	“Event End Time Updated Successfully” notification.
Expected output (Negative):	“Event item not found” notification.

Observed output:	The user is able to change the notes of an entry from the list. The updated notes are displayed to the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 39

Requirement name:	Change Event's Time Estimate
Test case description:	The users attempt to change the Time Estimate value of an entry from the To-do List.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account existing entry
Input:	User clicks on the To-do List, selects a task, and enters the new task time estimation.
Expected output (positive):	"Time Estimate Updated Successfully" notification.

Expected output (Negative):	“Entry item not found” notification. Invalid time format.
Observed output:	The user is able to change an event’s time estimate accordingly. The updated event is displayed to the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 40

Requirement name:	Change status of an entry from the To-do List
Test case description:	User attempts to change the status of an entry from the To-Do List.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account existing entry
Input:	User clicks on the To-do List, selects a task, and changes the status of an entry by selecting the appropriate label.
Expected output (Positive):	“Event status updated successfully” Notification.

Expected output (Negative):	“Event Item Not Found” Notification. Status choice not found in the list (In progress, Complete, Not Started).
Observed output:	The user is able to update the status of a task. The updated status is displayed to the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 41

Requirement name:	Mark an entry from the To-do List as “Done”.
Test case description:	The user attempts to mark an entry from the To-do List as “Done”.
Pre-requirements:	ernet connection to the application server ccessful login to the registered account, existing entry.
Input:	User clicks to the To-do List page, selects an entry to mark it as “Done”.

Expected output (Positive):	“Event Complete” Notification. (Could be dropped from list, or visually represent its completeness)
Expected output (Negative):	“Event item not found” Notification.
Observed output:	The user is able to mark an entry as “Done” once completed. The updated entry is displayed to the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 42

Requirement name:	Logout.
Test case description:	User attempts to logout from their account.
Pre-requirements:	ernet connection to application server g in to a valid account.
Input:	Click on Logout.

Expected output (positive):	User has successfully logged out from their account and is redirected to the Welcome page.
Expected output (Negative):	User is not logged in, or the session has expired.
Observed output:	The user is able to logout from their account and is redirected to the Welcome page.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 43

Requirement name:	Reset Password
Test case description:	User can change their account password if forgotten.
Pre-requirements:	ernet connection to application server

	gistered account in the database.
Input:	Navigate to Login page, click on reset password or the link, enter the email address associated with the user's account to change the password. Check the email inbox, choose another password.
Expected output (positive):	Successful password reset message.
Expected output (Negative):	The reset link has expired. Password doesn't meet the requirements.
Observed output:	The user is able to reset their password.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 44

Requirement name:	Linking an Inbox.
Test case description:	User links an existing email address inbox to the application using Oauth.
Pre-requirements:	ernet connection to application server g in to a valid account.
Input:	Navigate to the Settings page.

	Click on linking an email inbox. Login with the preferred email address to link it with the application.
Expected output (positive):	Successful email linking message.
Expected output (Negative):	Email inbox could not be authenticated error.
Observed output:	The user is able to link their inbox with the application.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 45

Requirement name:	To-Do-list statistics
Test case description:	Returns the statistics of completed, not started, and in-progress tasks of the day.
Pre-requirements:	ernet connection to application server er logged in to the account.

Input:	The application automatically fetches the user's To-do List's statistics.
Expected output (positive):	Completed tasks of the day. In-progress tasks of the day. Not started tasks of the day.
Expected output (Negative):	-
Observed output:	The user is able to view their progress of completed tasks from the To-do List
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Admin

Function 46

Requirement name:	Remove User
Test case description:	Administrator removing a user.

Pre-requirements:	ernet connection to application server g in to a valid administrator account.
Input:	Navigate to the administrator page, click on users, select the user to be delete, click on delete, then confirm deletion.
Expected output (positive):	Use was deleted successfully.
Expected output (Negative):	-
Observed output:	Admin successfully removed a user from the application.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 47

Requirement name:	Add User

Test case description:	Administrator adding a user.
Pre-requirements:	ernet connection to application server g in to a valid administrator account.
Input:	Navigate to administrator page, click on users, select user button, fill the credentials of the new user.
Expected output (positive):	User was added successfully.
Expected output (Negative):	Specific fields don't meet the requirements. Username or email address already exists.
Observed output:	Admin successfully added a user to the application.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 48

Requirement name:	Update User
Test case description:	Administrator updating user credentials.
Pre-requirements:	ernet connection to application server g in to a valid administrator account.
Input:	Navigate to the administrator page, click on users, select a user, change the user's credential then click on save.
Expected output (positive):	Use was changed successfully.
Expected output (Negative):	Specific fields don't meet the requirements. Username or email address already exists.
Observed output:	Admin successfully changes a user credential saved in the system.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Function 49

Requirement name:	Add/revoke permission.
Test case description:	Administrator updating user permissions by adding or revoking certain permissions.
Pre-requirements:	ernet connection to application server g in to a valid administrator account.
Input:	Navigate to the administrator page, click on users, select the user, add/revoke permissions, then save.
Expected output (positive):	Use was changed successfully.
Expected output (Negative):	-
Observed output:	Admin has successfully added/revoked a permission from the user.
Verdict:	Function is fully operational & passed the Acceptance testing phase.
Comments:	-

Test Results:

The Acceptance testing revealed:

An ample number of incomplete functions missed by the back-end team. Which after were covered and dealt with.

Functions with unimplemented error handling, which were addressed on the spot.

The need for further security implementation with code optimization on some functions which was completed.

Conclusion and Future Work

Conclusion

"Saturday" has proven to be quite the challenge to be completed within the time frame of the two trimesters. The scope of the project narrowed throughout the second trimester as reality had reared its ugly head. In conclusion, Saturday presents an AI Assistant designed to help young adults, particularly students and university professors, manage their time, projects, and communications more effectively. This innovative solution transcends conventional personal assistants by offering a cost-effective alternative that leverages advanced algorithms and data-driven insights for seamless integration into daily routines. Aimed at enhancing productivity and reducing stress, it addresses the challenges of utilizing modern digital tools—calendars, to-do lists, and emails—transforming these into opportunities for improved management and efficiency. Crafted through a user-centered design methodology, Saturday's AI Assistant is built on a technological foundation encompassing Application Design, Machine Learning, Datasets, Natural Language Processing (NLP), and Algorithm Development. Addressing issues such as cross-platform integration and data security, it provides a robust, scalable solution that exceeds the expectations of its target demographic, marking a significant advancement in personal productivity tools.

Most importantly, the technological underpinning of our research includes Natural Language Processing (NLP), Algorithm Development, Machine Learning, and Application Design. Moreover, the Double Diamond model and design thinking concepts have guided our user-centered and iterative design process, which has helped us tackle the challenges of creating a sophisticated tool that is both responsive and adaptable to the demands of its users. We have developed a system that is reliable, scalable, and easily interfaces with well-known email and calendar platforms. We are aware of the limitations and difficulties that come with undertaking a project of this nature, including concerns about data security and cross-platform interaction. Last but not least... the result is a strong, customized AI assistant that not only meets but goes beyond the needs of our target market, providing a guiding light amidst the ever-more-complex world of work and education.

Strengths and Weaknesses

The Saturday AI assistant project has a combination of strengths and limitations. Its ambitious goal is to improve productivity and stress management for university instructors and students using innovative technical solutions. First, let's look at our strengths:

1- Innovative Approach: Saturday uses algorithms and data-driven insights to provide personalized assistance in managing time, tasks, and communications, setting it apart from traditional personal assistant solutions.
2- Cost-Effectiveness: Designed as a more affordable alternative to personal or virtual assistants, it makes advanced productivity tools accessible to a broader audience, particularly benefiting those in academia.
3- User-Centric Design: The project's iterative design process, guided by user feedback and centered around user needs, ensures that the final product is intuitive, responsive, and truly useful for its target demographic.
4- Technological Foundation: With a strong emphasis on Application Design, Machine Learning, NLP, and Algorithm Development, Saturday is poised to offer a sophisticated solution that can adapt to and learn from individual user preferences and behaviors.

Our weaknesses:

1- Scope and Timeline Challenges: The project faced difficulties in meeting its initial scope within the planned two-trimester timeline, indicating potential underestimation of the complexities involved in developing such an advanced AI system.
2- Technological Constraints: While ambitious, the project may encounter limitations in current AI and NLP technologies, which could affect the AI assistant's ability to fully understand and predict user needs accurately.
3- Data Privacy and Security: Handling sensitive information, especially for an academic audience, requires stringent security measures. The project must navigate the fine line between personalized assistance and privacy concerns.
4- Adaptation and Learning Curve: Despite being designed for ease of use, there may still be a learning curve for users to fully utilize all features effectively. Additionally, the system's adaptability over time is crucial but not guaranteed.

In summary, Saturday represents a progressive response to common productivity issues, bolstered by a robust technological and user-centered base. It also must overcome major obstacles regarding user data security, technological constraints, integration difficulties, and project scope. For the project to succeed and satisfy its intended customers' demands, these flaws must be fixed.

Future Improvements

Saturday appears to be headed in a positive direction with lots of room for growth and improvement. One important aspect that must be improved is the Natural Language Processing

(NLP) capabilities. By improving natural language processing, Saturday will be able to provide more sophisticated and situation-aware interactions that closely resemble the comprehension and flexibility of a human helper. This will hone the assistant's communication abilities and enhance its capacity to handle intricate tasks and schedules, making it an even more valuable tool for its users. Further personalization of the user experience can be achieved by concentrating on the development of machine learning algorithms. Saturday can predict requirements and preferences by analyzing user behavior over time and providing customized recommendations and automations.

The development of integration capabilities with a wider range of platforms and services is another important area for advancement. Saturday has the potential to develop into a completer and more adaptable helper by forming alliances and being compatible with additional professional and educational tools. Deeper integration with project management software, academic software, and collaboration platforms are all part of this, making sure that users have a single point of contact for all their needs—both personal and professional. In addition, tackling the obstacles associated with cross-platform usability will improve accessibility, so transforming Saturday into a more comprehensive solution that serves a varied user population. Maintaining user trust and compliance will also require a constant focus on data security and privacy in accordance with the most recent standards and legislation.

References

- Anon, (2021). *New Data Protection Law introduced in the UAE – 10 key takeaways*. [online] Available at: https://www.herbertsmithfreehills.com/insights/2021-12/new-data-protection-law-introduced-in-the-uae-%E2%80%93-10-key-takeaways.
- u.ae. (n.d.). *Citizens' right to access government information | The Official Portal of the UAE Government*. [online] Available at: https://u.ae/en/about-the-uae/digital-uae/regulatory-framework/citizens-right-to-access-government-information.

- Law, P. (2023). *UAE Data Protection Law: Essential Guide*. [online] TR - Legal Insight MENA. Available at: https://insight.thomsonreuters.com/mena/legal/posts/uae-data-protection-law-essential-guide.
- Artificial Intelligence Office, UAE. (n.d.). *Artificial Intelligence*. [online] Available at: https://ai.gov.ae/.
- Artificial Intelligence Office, UAE. (n.d.). *AI Program*. [online] Available at: https://ai.gov.ae/ai-program/.
- u.ae. (n.d.). *Artificial intelligence | The Official Portal of the UAE Government*. [online] Available at: https://u.ae/en/about-the-uae/digital-uae/digital-technology/artificial-intelligence.
- Basheer, S. (2023). *Dubai Centre for AI's Accelerator Programs attract 615 startups from 55 countries*. [online] Dubai Future Foundation. Available at: https://www.dubaifuture.ae/latest-news/dubai-centre-for-ais-accelerator-programs-attract-615-startups-from-55-countries/.
- u.ae. (n.d.). *Data protection laws - The Official Portal of the UAE Government*. [online] Available at: https://u.ae/en/about-the-uae/digital-uae/data/data-protection-laws.
- cms.law. (n.d.). *Data protection and cybersecurity laws in UAE | CMS Expert Guide*. [online] Available at: https://cms.law/en/int/expert-guides/cms-expert-guide-to-data-protection-and-cyber-security-laws/uae.
- DataGuidance. (2020). *UAE - Data Protection Overview*. [online] Available at: https://www.dataguidance.com/notes/uae-data-protection-overview.
- U, C. (2022). *Keeping Up With Data Regulations In The UAE*. [online] Datatechvibe. Available at: https://datatechvibe.com/data/keeping-up-with-data-regulations-in-the-uae/.
- Peterdy, K. (2024). *PESTEL analysis*. [online] Corporate Finance Institute. Available at: https://corporatefinanceinstitute.com/resources/management/pestel-analysis/.
- Anon, (n.d.). *Class Responsibility Collaborator (CRC) Models: An Agile Introduction – The Agile Modeling (AM) Method*. [online] Available at: https://agilemodeling.com/artifacts/crcmodel.htm.